

PROFILE OF SOFTWARE at the INFORMATION SYSTEMS CENTER

DECEMBER 1999



National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

Foreword

The Software Engineering Laboratory (SEL) is an organization sponsored by the National Aeronautics and Space Administration/Goddard Space Flight Center (NASA/GSFC) and created to investigate the effectiveness of software engineering technologies when applied to the development of applications software. The SEL was created in 1976 and has three primary organizational members:

NASA/GSFC, Systems Integration and Engineering Branch

University of Maryland, Department of Computer Science

Computer Sciences Corporation, Development and Sustaining Engineering

The goals of the SEL are (1) to understand the software development process in the GSFC environment; (2) to measure the effects of various methodologies, tools, and models on this process; and (3) to identify and then to apply successful development practices. The activities, findings, and recommendations of the SEL are recorded in the Software Engineering Laboratory Series, a continuing series of reports that includes this document.

The following are the primary contributors to this document:

Howard Kea, Goddard Space Flight Center

Steve Kraft, Goddard Space Flight Center

Mike Stark, Goddard Space Flight Center

Vic Basili, University of Maryland

Jeff Carver, University of Maryland

Maurizio Morisio, University of Maryland

Carolyn Seaman, University of Maryland

Richard Webby, University of Maryland

Daniil Yakimovich, University of Maryland

Jackie Boger, Computer Sciences Corporation

Steve Condon, Computer Sciences Corporation

Linda Landis, Computer Sciences Corporation

Amy Parra, Computer Sciences Corporation

David Schultz, Computer Sciences Corporation

Documents from the Software Engineering Laboratory Series can be obtained via the SEL homepage at

<http://sel.gsfc.nasa.gov>

or by writing to:

Systems Integration and Engineering Branch
Code 581
Goddard Space Flight Center
Greenbelt, Maryland, U.S.A. 20771

Acknowledgements

The following GSFC civil servants and contractor personnel participated directly in this study.

Participant	Organization	Participant	Organization
Leslye Boyce	581	Dan Mandl	584
Steve Tompkins	581	Eve Rothenberg	584
Nate Wright	581	Tom Taylor	584
Bob Kozon	581	Susan Semancik	584
Steve Coyle	581	Michael Matthews	584
Bill Decker	581	Howard Eisericke	585
Elaine Shell	582	Steve Naus	585
Lisa Shears	582	Dennis Giblin	585
Ray Whitley	582	Owen Kardatzke	585
Rich Heasty	582	Mary Anne Esfandiari	586 and 587
Kris Naylor	582	Mary Reph	586
Phil Myers	582	Joy Henegar	586
Adrian Hill	582	Robert Schweiss	586
Bruce Savadkin	582	Jeannine Shirley	586
Manuel Maldonado	582	Jim Byrnes	587
Jan Owings	582	Igor Eberstein	587
Tina Fredo	582	Doug McCuistion	588
Carlos Trujillo	582	Julie Breed	588
Mike Blau	582	Karl Mueller	588
Jonathan Wilmot	582	Jeremy Jones	588
Henry Murray	583	Tom Grubb	588
Scott Green	583	Walt Truszkowski	588
Sally Godfrey	583	Jeff Lubelczyk	588
Carla Matusow	583	Troy Ames	588
Rodger Abel	583	Matthew Brandt	588
Barbara Pfarr	584	Johnny Medina	588
John Donohue	584	Barbara Brown	588
Jay Pittman	584	Jeffrey Hosler	588
Barbara Milner	584	Walt Moleski	588
Karen Keadle-Calvert	584	James Rash	588
Pam Pittman	584	Mark Stirling	588
Ken Lehtonen	584	George Seftas	588
Mark Rice	584	Dana Uehling	588
Patrick Hennessey	584	Susan Valett	588
Jeffrey Ferrara	584	Nigel Ziyad	588

Executive Summary

This document presents a profile of the Information Systems Center (ISC), Code 580, at the National Aeronautics and Space Administration (NASA) Goddard Space Flight Center (GSFC). *The information for this profile was gathered between July 1998, and August 1999.* This report was prepared by the Software Engineering Laboratory (SEL), which is a group within the ISC that studies software development projects in order to help improve software products and processes. The information presented in this report was compiled primarily through interviews with ISC branch heads and team leads, and from questionnaires that these branch heads and team leads completed and submitted to the SEL.

The purpose of this profile is to present a snapshot of the entire ISC organization. It summarizes the work being performed, the organization of the various branches, the software products being developed or maintained, and the software processes being employed. It is hoped that the profile will help ISC management understand the makeup of the Center and identify candidate areas for further process improvement. It is also hoped that this profile will provide a point of comparison for tracking future changes and improvements. To this end, it is anticipated that a subsequent profile will be prepared in 3-5 years to permit the ISC to track its evolution.

Key observations regarding the personnel within the ISC:

- Much of the work of the ISC is performed by teams, which typically comprise between 3 and 8 persons.
- ISC team leads typically have between 3 and 10 years of project management experience. The experience range varies significantly from branch to branch, however, and many team leads fall outside the typical range.
- Contractor personnel make up a significant component of many ISC teams. Nearly half of the “typical” teams surveyed comprised more contractor personnel than civil servants.
- A significant number of ISC personnel are matrixed to organizations outside the ISC (e.g., Codes 400, 600, or 900), or, in a few cases, to other branches within the ISC.
- On the average, ISC software personnel spend about 7.4 days per year in software-related training (primarily on-the-job training [OJT]).

Key observations regarding ISC software products:

- ISC is supporting both development of new software systems and maintenance of existing software systems, but more effort is going into development than into maintenance.
- It is difficult to put bounds on the volume of software being developed or maintained within ISC, because the various ISC branches and teams use different methods for measuring and capturing the size of their software systems.
- The operational lifetime of ISC software is typically between 2 and 7 years, although systems with operational lifetimes both below and above this range were reported.
- There is a pronounced trend to move away from legacy languages such as FORTRAN, C, and assembly, to newer languages such as Java and Perl.
- ISC teams report a significant use of commercial off-the-shelf (COTS) products as components of deliverable systems (embedded COTS), as software support tools delivered with the system, and as undelivered support tools.

Key observations regarding ISC software processes:

- Software requirements appear to be reasonably stable among ISC projects. Over half the teams reported that their requirements were either “fairly stable” or “very stable.”
- The use of software processes and standards varied considerably among branches and projects, with most processes and standards being defined at the project, rather than the branch, level.
- On the average, 3.5% of the annual budget of the ISC software teams is allocated to software engineering research.
- Documentation serves as a primary means of communication among the members of ISC software project teams. The documents most often cited as being produced are, in decreasing order of frequency: User’s Guide, Requirements Specification, Design Document, Test Plan, Project Plan, and Configuration Management Plan.

- There is evidence that some of the ISC project teams that perform software maintenance are using standard software maintenance practices to protect software product integrity. Of these standard maintenance practices, only regression testing is consistently employed among the teams surveyed.
- There is significant involvement in software process improvement within the ISC, and a variety of process improvement activities were reported. There is, however, no evidence of an integrated process improvement effort at the Center level.
- Collection of software metrics is still the exception rather than the norm among ISC projects. Only seven of the 20 teams surveyed reported that they routinely collect and analyze at least one kind of software metric.

Table of Contents

1	Introduction.....	1-1
1.1	The Critical Importance of Software to Goddard.....	1-1
1.2	Goddard’s Center for Information Systems.....	1-1
1.3	NASA’s Software Engineering Laboratory	1-3
1.4	Role of this Profile	1-4
1.5	Definitions	1-5
1.6	Document Organization.....	1-6
2	Methodology	2-1
2.1	Scope	2-1
2.2	Data Collection and Analysis Approach.....	2-1
2.2.1	Interview Guides	2-2
2.2.2	Questionnaires	2-2
2.2.3	Verification and Analysis	2-2
3	Sample Universe: Branches and Teams.....	3-1
3.1	Characterization of Branches	3-1
3.1.1	Management: Code 581.....	3-1
3.1.2	Mission-Directed Software Development: Codes 582, 583, 584, and 586.....	3-1
3.1.3	Advanced Technology: Codes 585, 587, and 588.....	3-1
3.2	Characterization of Teams by Function.....	3-1
3.2.1	Management Functions	3-1
3.2.2	Software Development for Operational Use.....	3-2
3.2.3	Software Development for Research Purposes	3-2
3.2.4	Small teams.....	3-2
3.3	Characterization of Teams by Team Size and Type of Software Activity	3-2
3.3.1	Size of Team	3-2
3.3.2	Division of Software Effort Between Development and Maintenance Work.....	3-5
3.4	Other Software-Related Teams Within the ISC.....	3-8
3.4.1	Sizes of Software-Related Teams	3-8
3.4.2	Division of Software Effort Between Development and Maintenance	3-12
4	Software Product Characteristics	4-1
4.1	Domains.....	4-1
4.2	Amount of Operational Software	4-2
4.3	Development Languages	4-2
4.4	COTS Implementation.....	4-5
4.5	Operational Lifetime of Software.....	4-8

4.6	Software Error Characteristics	4-8
4.7	Requirements Stability	4-8
5	Software Process Characteristics.....	5-1
5.1	Software Processes and Standards	5-1
5.1.1	Software Development Processes and Standards.....	5-1
5.1.2	Maintenance Processes and Standards	5-1
5.1.3	Documentation Standards.....	5-2
5.2	Project Management Practices	5-3
5.2.1	Management experience	5-3
5.2.2	Matrixing	5-4
5.3	Software Engineering Practices	5-4
5.3.1	Development Methodologies	5-4
5.3.2	Testing Methods.....	5-6
5.3.3	Software IV&V	5-8
5.3.4	Development Tools	5-8
5.3.5	Software Reuse	5-9
5.3.6	Maintenance Practices.....	5-10
5.4	Other Issues.....	5-11
5.4.1	Software Training	5-11
5.4.2	Software Measures	5-12
5.4.3	Software Engineering Research	5-12
5.4.4	Process Improvement Activities	5-12
6	Recommendations.....	6-1
6.1	Recommendations	6-1
6.1.1	Training	6-1
6.1.2	COTS	6-1
6.1.3	Processes and Standards	6-1
6.1.4	Project Plans	6-2
6.1.5	Metrics	6-2
6.2	Other Suggestions	6-2
6.2.1	Process Definition	6-2
6.2.2	Process Improvement Initiatives	6-3
6.2.3	ISO 9001 Registration	6-3
6.3	Areas for Further Study	6-3
Appendix A: ISC Baseline Branch-Level Interview Guide		A-1
Appendix B: ISC Baseline Branch-Questionnaire.....		B-1
Appendix C: ISC Baseline Team-Level Interview Guide		C-1
Appendix D: ISC Baseline Team-Level Questionnaire.....		D-1
Appendix E: ISC Software System Development Team Questionnaire.....		E-1
Abbreviations and Acronyms.....		AB-1
References		R-1
Standard Bibliography of SEL Literature		BI-1

Figures

Figure 1. Organization of the Information Systems Center (ISC).....	1-2
Figure 2. SEL Experience Factory.....	1-4
Figure 3. Number of ISC Teams Studied.....	2-1
Figure 4. Breakdown of FTEs by Team.....	3-3
Figure 5. Breakdown of FTEs by Team Without the Two Largest Teams.....	3-3
Figure 6. Breakdown of FTEs per Typical Team.....	3-4
Figure 7. Distribution of Effort Between Development and Maintenance (Typical Teams).....	3-5
Figure 8. Distribution of Development Effort By Activity (Typical Teams).....	3-6
Figure 9. Distribution of Maintenance Effort by Activity (Typical Teams).....	3-7
Figure 10. Allocation of Maintenance Effort by Type of Maintenance (Typical Teams).....	3-7
Figure 11. Civil Servant and Contractor FTEs on Software Teams (Codes 581-586).....	3-9
Figure 12. Civil Servant and Contractor FTEs on Software Teams (Codes 581-586), Leaving out the Two Largest Teams.....	3-10
Figure 13. Civil Servant and Contractor FTEs on Software Teams (Code 588).....	3-11
Figure 14. Distribution of Effort Between Development and Maintenance (Codes 581-586).....	3-12
Figure 15. Distribution of Effort Between Development and Maintenance (Code 588).....	3-13
Figure 16. Programming Languages Used in Development as Compared to Maintenance.....	4-4
Figure 17. Development Languages Used Within Each ISC Domain.....	4-4
Figure 18. Key Documents Produced.....	5-3
Figure 19. Comparison of Team Responses (Averaged by Branch) Versus Branch Manager's Responses for Awareness, Training and Usage of Nine Development Methodologies.....	5-5
Figure 20. Use of Routinely Applied Tools.....	5-9
Figure 21. Percent of Reused Code.....	5-10
Figure 22. Use of Standard Maintenance Practices.....	5-11

Tables

Table 1. ISC Branches and their Missions.....	1-3
Table 2. Statistics for Civil Servants and Contractor Personnel in 48 ISC Software Teams.....	3-8
Table 3. Number of ISC Teams Working in each Domain.....	4-1
Table 4. Domain Descriptions and Types of Software.....	4-2
Table 5. Use of Programming Languages.....	4-3
Table 6. Use of Embedded, Delivered, and Non-Delivered COTS Products.....	4-6
Table 6 (cont.). Use of Embedded, Delivered, and Non-Delivered COTS Products.....	4-7
Table 7. Reported Use and Helpfulness of Software Processes and Standards.....	5-2
Table 8. Leadership Experience of Team Leads, by ISC Branch.....	5-4
Table 9. Use of Testing Methods in ISC Teams.....	5-7
Table 10. Collection of Metrics among the ISC Branches.....	5-12

1 Introduction

1.1 The Critical Importance of Software to Goddard

On May 1, 1959 the fledgling National Aeronautics and Space Administration (NASA) established its first space flight center in Greenbelt, Maryland, a few miles from Washington, D.C. NASA named the Center after American rocket Pioneer Robert Hutchings Goddard. Today Goddard Space Flight Center (“Goddard” or GSFC) is home to the nation's largest organization of combined scientists and engineers dedicated to learning and sharing their knowledge of the Earth, solar system, and universe. Goddard enables discovery through leadership in Earth and space science. Its mission is to serve the scientific community, inspire the nation, foster education, and stimulate economic growth. It does this by partnering with others to achieve NASA's goals. The Goddard community of scientists and engineers creates technologies that support and advance these endeavors to take full advantage of space research.

Since 1959, the number of computers used at Goddard and the number of lines of code that Goddard maintains have grown astronomically. By 1993, Goddard had an inventory of 40 million source lines of code (MSLOC). Approximately 25% of Goddard's 12,500 civil servants and contractors were required to develop, assure, manage, and maintain Goddard software (Reference 1). These numbers demonstrate both the importance of software to Goddard's mission as well as the high cost of software.

The situation across all NASA centers closely mirrors the situation at GSFC. According to a 1995 NASA study, approximately \$450 million of NASA's FY93 budget was required to maintain NASA's 160 MSLOC. Another \$550 million were spent that year developing an additional 6 MSLOC. More than 10 percent of NASA's civil service and contractor workforce spent the majority of their time managing, developing, assuring, and/or maintaining software (Reference 2).

In the years since this 1995 report, the reliance of NASA—and American technology as a whole—on software has only increased. The *President's Information Technology Advisory Committee* consists of CEOs and vice presidents of major information technology corporations, directors of major government and academic laboratories, and renowned professors in the information sciences. In August 1998 this committee issued its interim report to the President, in which it claimed, “Vigorous information technology research and development (R&D) is essential for achieving America's 21st century aspirations.” The report characterized software as the “infrastructure of the information age” and claimed that software was “fundamental to economic success, scientific and technical research, and national security.” At the same time, however, the authors pointed out that the demand for software had outstripped the ability to produce it.

“The result is that desperately needed software is not being developed. Furthermore, the Nation currently depends on software that is fragile, unreliable, and extremely difficult and labor-intensive to develop, test, and evolve. Our ability to construct the needed software systems and our ability to analyze and predict the performance of the enormously complex software systems that lie at the core of our economy are painfully inadequate” (Reference 3).

1.2 Goddard's Center for Information Systems

The importance of software to Goddard's strategic mission and institutional goals was recognized when a major reorganization of Goddard divisions and directorates took place at the beginning of 1998. A new “Information Systems Center” (ISC) was created with the objective of concentrating and consolidating most of Goddard's information technology (IT) capabilities into one organizational unit. IT received yet more stature in December of the same year, when Goddard Director Al Diaz announced at the 23rd Software Engineering Workshop that IT would be one of the “core disciplines” at GSFC and that he would “put it at the top of the list.”

As a result, the ISC is Goddard's core of expertise in the implementation and management of information systems supporting GSFC missions. The ISC collaborates with the Earth and space science communities, project management and other customers to meet their information technology needs through the design, implementation, and integration of information systems and system components. Additionally, the ISC provides leadership and vision in identifying and sponsoring new and emerging information systems technologies.

The ISC consists of over three hundred civil servants, of whom approximately seventy percent were drawn from the Mission Operations and Data Systems Directorate (Goddard Code 500). Another ten percent of the ISC staff were drawn from the Engineering Directorate (Goddard Code 700). Both of these directorates became defunct with the GSFC reorganization. The remaining twenty percent of ISC staff were originally from Goddard Codes 200, 300, 400, 600, 800, and 900.

The ISC is organized into eight branches and one office, as shown in Figure 1. This chart and other information about the ISC can be found at the ISC web site, <http://isc.gsfc.nasa.gov>.

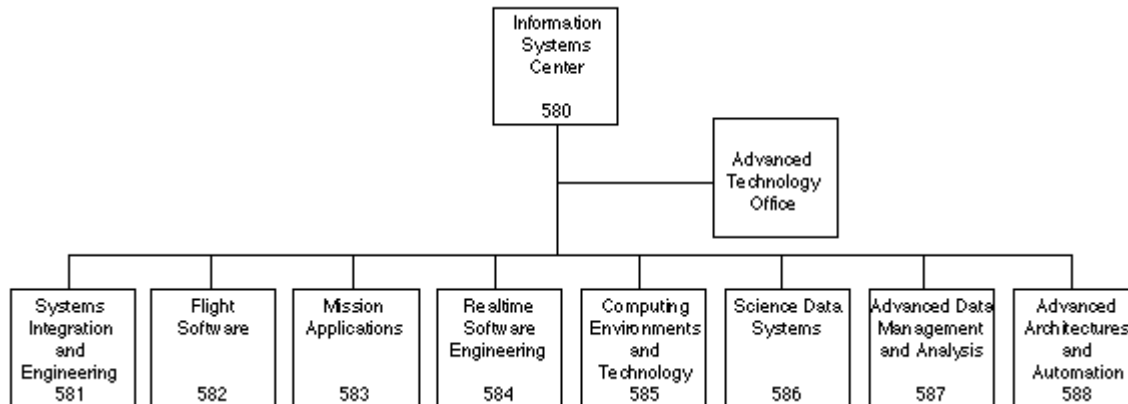


Figure 1. Organization of the Information Systems Center (ISC)

The ISC mission is to provide high value information systems products, services and expertise, and to advance information technologies that are aligned with customer needs.

The ISC vision is to be a world-class information systems center of excellence serving the needs of GSFC and NASA customers.

Strategic goals established by the ISC are:

- Advance leading-edge information system technology;
- Clearly define the scope of ISC business, and deliver high value products and services that satisfy customer needs;
- Build a diverse, talented, innovative, energized, internationally recognized workforce of employees and managers; and
- Establish open, flexible, collaborative relationships with customers and partners.

The missions of the individual ISC branches are shown in Table 1.

Table 1. ISC Branches and their Missions

Branch Code	Branch Name	Branch Mission
581	Systems Integration and Engineering Branch	Provides expert advice and technical consultation to Principal Investigators and Mission Study Managers on operational concepts, data systems architectures, life cycle costing, software process improvements, technology assessments and product evaluation. Provides end-to-end engineering of ISC mission systems development activities, including software development and flight operations.
582	Flight Software Branch	Provides end-to-end life cycle products associated with embedded software for spacecraft, scientific instruments, and flight components.
583	Mission Applications Branch	Provides for the development of off-line systems and applications to support Earth and Space Science missions. Develops operational mission data systems that include functions such as science planning and scheduling aids, guidance navigation and control software, and Network Control Center Data systems.
584	Real-Time Software Engineering Branch	Develops ground data systems for integration and test and on-orbit operations of Earth and Space Sciences Missions.
585	Computing Environments and Technology Branch	Provides infrastructure support to scientific, project, administrative and outreach activities across GSFC. Branch personnel collaborate with various elements at GSFC to meet their information system needs in WWW application development and database design, network system engineering, IFMP system engineering (Agency level), and facility support.
586	Science Data Systems Branch	Develops systems for operational data capture, level zero and higher level data processing, data archival, data distribution, data analysis, information management, and science data system proposal development support. These systems may include various levels of science and telemetry data processing, starting from the point of the data reaching the ground to their delivery to scientific users for analysis.
587	Advanced Data Management and Analysis Branch	Develops systems that address data display, data analysis, data visualization, data archiving, and mass storage. Provides support for algorithm development, science data analysis programming, data mining, data retrieval, fusion and dissemination, scientific mission proposal development, and support for large and small-scale software system configuration, sizing, and development methodologies.
588	Advanced Architectures and Automation Branch	Explores, develops, and promotes state-of-the-art software and networking technologies critical for improving the effectiveness, and reducing the costs, of future generations of mission data and information systems. Works primarily through collaborations with other NASA and government organizations, universities, and commercial partners.

1.3 NASA's Software Engineering Laboratory

The GSFC reorganization that concentrated most of Goddard's software maintainers, developers, and experimenters into the newly formed ISC also brought into the ISC a consortium of software process and product improvement specialists known as the NASA Software Engineering Laboratory (SEL) (Reference 4). At twenty-three years old, the SEL is both one of the oldest software process improvement laboratories in the world, as well as one of the longest standing examples of partnering in existence at Goddard.

The SEL was created in 1976 at NASA Goddard for the purpose of understanding and improving the overall software process and products of the Flight Dynamics Division (FDD). A partnership was formed between the FDD, the Computer Sciences Department of the University of Maryland (UM), and Computer Sciences Corporation (CSC). Each partner played a key role. The FDD provided the user, manager, and NASA perspective. The UM supplied academic rigor to experimentation and introduced some of the latest concepts in software processes. CSC, as the major contractor responsible for building and maintaining the flight dynamics software, furnished the "real-world engineering" or "craftsman" perspective.

In order to carry out process improvements within the FDD, the SEL defined and implemented an organizational concept named the *Experience Factory* (Reference 5). The experience factory institutionalizes the collective learning of the organization that is the root of continual improvement and competitive advantage. It establishes a separate organizational element that is responsible for collective learning and technology transfer activities. The job of the experience factory is to record, analyze, formalize, tailor, and synthesize the experiences of the project organization that is responsible for developing and maintaining software. This structure creates a symbiotic relationship where the

- Project organization offers to the experience factory its products, the plans used in its development, and the data gathered during development and operation.
- Experience factory staff transform these objects into reusable units and supply them to the project organization, together with specific support that includes monitoring and consulting.

As the project organization goes about its business of developing applications, the experience factory collects metrics and lessons learned. The experience-factory staff store these data in a database, analyze the data, and suggest and conduct experiments. They then package these distilled project experiences into recommended best practices, estimation models, and software development training courses, which spread these process improvements throughout the project organization. Figure 2 depicts this relationship between the project organization and the experience factory. A heavy dashed line separates the two groups.

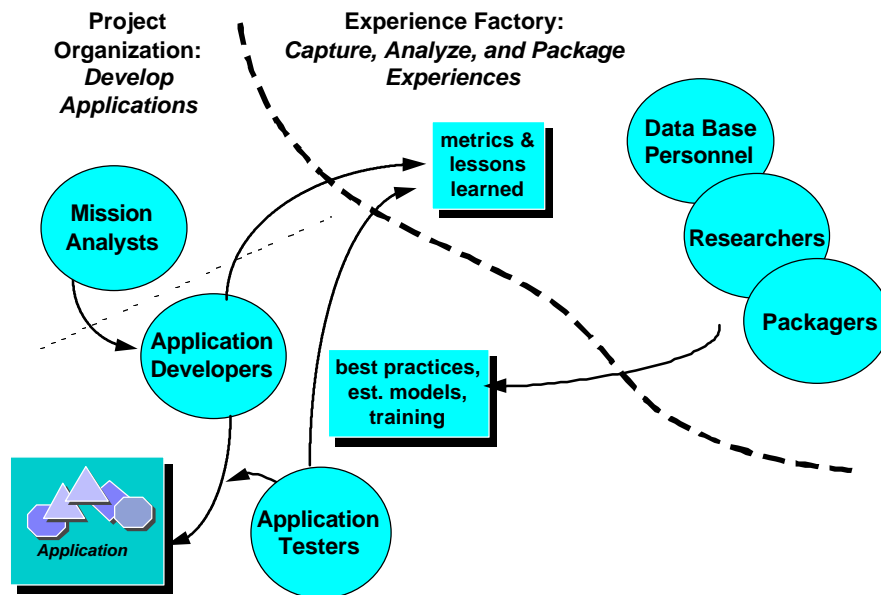


Figure 2. SEL Experience Factory

1.4 Role of this Profile

The first step to any improvement program is to understand the domain's current products, the process used to create those products, and the rationale and history behind that process. Without this understanding one cannot choose wisely how and where to make process improvements, and one cannot subsequently demonstrate that process *changes* have resulted in actual *improvements*. The information contained in this baseline must encompass the software process, product, and environment. Examples are as follows:

Software Process

How is software produced?
What standards are used?
What tools are used?

Software Products

How much software is produced?
What are the quality characteristics?

Environment

What is the organizational structure?
What training is available?
What research is underway?

The goals of this profile are:

- To assist ISC upper management in understanding the products that the ISC produces and the processes by which these products are created;
- To allow ISC branch heads to better understand activities in each branch of the ISC, thus stimulating cross-fertilization and technology transfer;
- By documenting the current products and processes of the ISC, to serve as a resource when the ISC chooses to institute process improvements or to make changes in response to reduced budgets or new requirements;
- To function as a benchmark against which to measure individual teams, branches, or the entire ISC at any point in the future.

1.5 Definitions

Baseline—A published specification of a profile (q.v.).

Branch—One of the eight components of the Information Systems Center (Code 580 at NASA/GSFC).

COTS—A commercial off-the-shelf software application. For the purposes of this ISC Profile, COTS is interpreted to include both delivered COTS and embedded COTS. COTS tools that are used for stand-alone support, however, are specifically excluded.

Domain—A specific field of activity or a particular area of knowledge, e.g., the flight software domain.

Matrixing—Assignment of a staff member from one ISC branch (q.v.) to support work being performed either within another ISC branch, or within another GSFC directorate. The directorates to which ISC personnel are most frequently matrixed are codes 400, 600, and 900.

Mission—(1) The special duty for which a group exists; (2) The particular task for which an artificial satellite is built, launched, and operated during its lifetime.

Object-Oriented Design—A software development technique in which a system or component is expressed in terms of objects and connections between those objects. (IEEE Std 610.12-1990, IEEE Standard Glossary for Software Engineering Terminology)

Object-Oriented Language—A programming language that allows the user to express a program in terms of objects and messages between those objects. (IEEE Std 610.12-1990, IEEE Standard Glossary for Software Engineering Terminology)

Profile—A detailed characterization of software and software engineering practices within a specified organization.

Project—A software project (q.v.). At GSFC ‘Project’ is used to mean the mission project.

Prototyping—A hardware and software development technique in which a preliminary version of part or all of the hardware or software is developed to permit user feedback, determine feasibility, or investigate timing or other issues in support of the development process. (IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology)

Reuse—The act of using on one software development project something (usually a code module) that was created for (and used on) an earlier software development project or created for a specific “reuse library.” Reused code can be used without modification, with slight modification, or with extensive modification. An object-oriented reuse library contains generalized objects that must be instantiated to create the specific code modules adopted for a given software development project.

Software Metrics—A quantitative measure of the degree to which a system component or process possesses a given attribute.

Software Process—A specification of the phases, activities, and products by which software is defined, developed, documented, and delivered. A process may provide for the use of standards, reviews, and metrics.

Software Process Improvement—The continual and iterative improvement of both the software process and products through the use of project experiences. (Software Process Improvement Guidebook, NASA-GB-001-95)

Software Project—The set of all project functions, activities, and tasks, both technical and managerial, required to satisfy the terms and conditions of the project agreement. A software project may be self-contained or may be part of a larger project. A software project may span only a portion of the software product lifecycle. (IEEE Std 1058.1-1987, IEEE Standard for Software Project Management Plans)

Software Standard—A specification of the format and content of a software product or document.

Team— A group of individuals working together for a joint purpose.

Training Program—A reasonably integrated set of related courses offered on a regular basis and designed to maintain and improve the skills necessary to develop, manage, assure, and deliver quality software using modern, proven techniques. (Profile of Software at NASA: NASA-RPT-04-95)

Typical Team—A team of people within the ISC that contains at least three members and develops or maintains software for operational use. (See Section 3.2.2 for further information).

Verification and Validation—A system engineering process employing a variety of software engineering methods, techniques, and tools for evaluating the correctness and quality of a software product throughout its life cycle. Verification is the process of determining whether or not the products of a given phase of the software development cycle fulfill the established requirements. Validation evaluates software at the end of the development lifecycle to ensure that the product not only complies with the specific criteria set forth by the customer, but performs as expected.

1.6 Document Organization

Chapter 2 discusses the characterization methodology for this study. It addresses the teams surveyed and the data collection and analysis methods that were employed. Chapter 3 describes the sample universe that was the object of the study. It provides a characterization of the branches and teams that were interviewed. Chapter 4 focuses on software products and the more quantifiable characteristics of the software across the ISC, addressing such topics as domains, development languages, COTS implementation, and software error characteristics. Chapter 5 discusses characteristics of the software processes within the ISC. It covers software processes and standards, project management practices, software engineering practices, and process improvement. Chapter 6 presents the process improvement recommendations of the study team, including a list of areas for further study.

Throughout Chapters 3 through 5, the results of this study are presented factually as objective data. The conclusions that the study team derived are summarized in Chapter 6.

2 Methodology

2.1 Scope

This study examines the products, processes, and environment of the ISC (GSFC Code 580). For this profile the SEL interviewed and collected detailed questionnaires from managers of all eight ISC branches and from selected team leaders using the process described in Section 2.2. Because of the large number of teams and time constraints for the study, team information was sampled. SEL staff interviewed the team leads for 26 (18 %) of the 145 ISC teams. Twenty of these 26 teams, or 14 % of all teams, returned the detailed team questionnaire.

Figure 3 shows the number of teams in each branch that were “interviewed & returned [a] questionnaire,” were “interviewed only,” or were not interviewed and returned no questionnaire. The branches with the largest percentage of teams supplying questionnaires were Code 583 and Code 585, where 25 % (two out of eight) of the teams supplied questionnaires.

The teams that we contacted were chosen by the branch heads to be representative of their branch. Some of these teams do not actually develop or maintain software, and therefore many of the questions on our questionnaire did not apply to them. Nevertheless, we reported the information collected from these teams, since it does help complete our snapshot for the present composition and workings of the ISC.

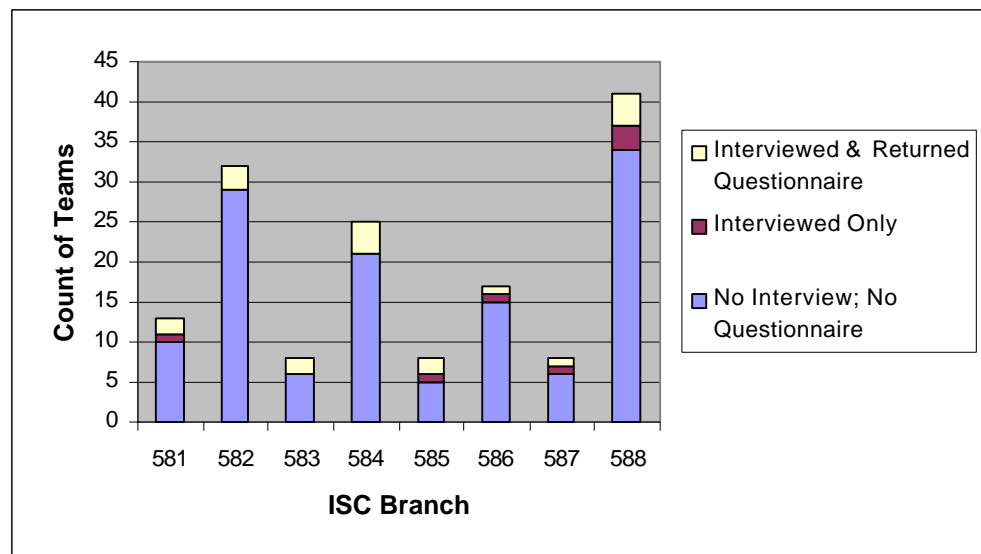


Figure 3. Number of ISC Teams Studied

2.2 Data Collection and Analysis Approach

The data collection approach was patterned after the method used in the NASA Software Engineering Program profiles of Goddard Code 500 (Reference 6), the Goddard Space Flight Center (Reference 1), and NASA (Reference 2). The techniques were tailored, however, based on lessons learned in surveys conducted for the SEL’s recent reuse and COTS studies. Reviewing the ISC web pages provided a high-level introductory understanding of the Center’s function, purpose, and organization.

To collect information about ISC processes, products, and people, the baseline study team utilized the techniques of structured interviews and questionnaires. We developed interview guides and questionnaires appropriate for each level. We gathered data at both the branch level and the team level. Following each branch or team interview, we asked the interviewee to complete the questionnaire and return it to us.

Interview guides were used to gather the qualitative data (e.g., expectations of future change), and questionnaires were used to collect the quantitative data (e.g., staff numbers). The aim was to characterize the software products, processes and people within the organization, with adequate qualitative context to meaningfully interpret the hard quantitative data.

2.2.1 Interview Guides

The face-to-face interviews with the branch management teams were the first mode of data collection. The SEL team then interviewed team leads whose names the respective branch heads had provided. Both the branch- and team-level interviews were designed to take no more than 30–45 minutes and cover the qualitative data. The process used for both the branch- and team-level face-to-face interviews is described below.

During the interview, the *interviewer* asked questions following the outline of the *interview guide*. The *scribe* wrote notes and employed a tape recorder, if acceptable to the *interviewee*, to aid in preparation of the *interview report*. The interviewee was told that the interview report would not be considered final until the interviewee had read and approved it. At the end of the interview, the interviewer gave the interviewee a copy of the *questionnaire*, which asked questions of a more detailed, numeric nature, and requested that the questionnaire be completed and returned. The interviewer also scheduled a follow-up interview to clarify any areas of confusion or misunderstanding on either side.

After the interview, the scribe prepared the interview report summarizing the interviewee's responses to the questions on the standard interview guide. The interviewer then reviewed the report and sent it to the interviewee for concurrence. Once the interviewee's concurrence was received, the report was considered approved. The actual interview guides used for both the branch and the team are contained in Appendices A and C.

2.2.2 Questionnaires

Questions requesting data that might take time for the interviewee to gather were relegated to the questionnaire. Interviewees filled out the questionnaire to the best of their ability, following the guidance that estimating the data was preferred to extensive investigation to calculate the data. They were asked to return the completed questionnaire within two weeks of the initial interview. A follow-up interview was held to review the completed questionnaire and resolve any areas where either the question wasn't clear to the interviewee or the response was unclear to the interviewer. The result of the follow-up interview was a completed questionnaire that was understood by both the interviewer and the interviewee.

The actual branch and team questionnaires are contained in Appendices B and D.

2.2.3 Verification and Analysis

In research as in software development, errors that are caught and corrected early in the process are less expensive to correct than those caught later. Information gathered during interviews is subjective by nature, consisting of spoken text, which can sometimes be ambiguous when set down in print. For this reason, two-person teams consisting of an interviewer and a scribe were sent to conduct the interview, as described above. The study team also sought to verify the information collected both in interviews and in questionnaires, primarily by revisiting the personnel who furnished the data to clarify unclear or inconsistent information.

Additional verification was performed during the analysis stage. After the interview and questionnaire data were tabulated and graphed, the resulting charts and tables were compared to ensure completeness and consistency. In some cases, the data had to be reorganized and re-graphed to ensure that outlying data points did not skew or obscure results. The study team also returned to the branch personnel to obtain clarification or additional data.

Feedback was also obtained during the compilation of interim reports at both the branch and team levels. These interim reports contained intermediate analysis and data organized by branch and team. The reports were submitted to the appropriate set of interviewees (branch or team) for final review before being delivered to the ISC branch heads.

In one instance, the verification process spawned an additional data collection effort. When ISC management questioned whether the FTE numbers in an interim report were representative of the ISC as a whole, an additional questionnaire was developed to canvass the entire Center. The results of both the earlier and later analyses are presented in Sections 3.3 and 3.4.

3 Sample Universe: Branches and Teams

In July and September 1998, the SEL study team met with representatives from every branch within the ISC. Between August 1998 and February 1999, the team also met with representatives of selected teams within each branch. After initial analysis of the data, the study team sent out an additional questionnaire on staffing to the 48 ISC teams who were performing software-related work. These final questionnaires were completed in August 1999.

3.1 Characterization of Branches

The eight branches within the ISC fall into three distinct classes, as described below.

3.1.1 Management: Code 581

The work of Code 581, the Systems Integration and Engineering Branch, is unique in that it involves coordination and oversight. In its managerial role, Code 581 does not develop code, but performs work that influences software development. Mission management, end-to-end mission planning, and system engineering fall within the scope of Code 581.

3.1.2 Mission-Directed Software Development: Codes 582, 583, 584, and 586

Several branches are involved in software development and maintenance for operational use, on current or future missions. These include Code 582, Flight Software; Code 583, Mission Applications; Code 584, Real-time Software Engineering, and Code 586, Science Data Systems. Two of the unique features of this type of work are that it is driven by mission schedules and it requires response to an external customer.

3.1.3 Advanced Technology: Codes 585, 587, and 588

Some branches of the ISC develop software that is not intended for mission use. In Code 585, Computing Environments and Technology, web development is taking place. Code 587, Advanced Data Management and Analysis, is developing prototypes. Code 588, Advanced Architectures & Automation, is developing and evaluating new concepts and technologies. These three branches perform work that can be designated as “Advanced Technology.” The teams in these three branches are building a knowledge base for the future in which these methods and products are likely to support operational missions. As such, control of this work lies predominantly with the development team; there are few imposed schedules until a product is selected for a mission. Because these teams are serving R&D functions and not developing code for operational use, formal development processes are less critical to them.

3.2 Characterization of Teams by Function

It was also logical to classify the teams studied into four groups:

- Teams performing management functions (3 teams)
- Teams developing software for operational use (14 teams)
- Teams developing software for research purposes (4 teams)
- Very small teams (1-2 persons) (2 teams)

3.2.1 Management Functions

The teams performing management functions are, as a rule, those within Code 581. As noted above, Code 581 doesn't perform any actual development. It manages projects, and frequently draws upon resources from other GSFC codes to accomplish the work. We interviewed three people whom the 581 branch head designated as team leads within Code 581. Two of these team leads are actually project managers. One of them, in fact, manages a project so large that other branches within the ISC perform portions of the work. For the purposes of this study, we have considered that the software development on this project is being performed by the other ISC branches involved rather than by Code 581. The third team lead is actually a management consultant and, again, performs no significant software development. This team delivers products such as operations concepts, architectures, trade studies, schedules, and cost estimates, but develops no deliverable applications software.

3.2.2 Software Development for Operational Use

Roughly half of the teams interviewed are engaged in developing software for operational use. Generally, these teams are supporting GSFC projects in another directorate (Code 400, 600, or 900). In some cases, the team leads are matrixed to those projects (see Section 5.2.2, “Matrixing”). Twelve of these teams contain between 3 and 20 members. We shall refer to these 12 teams as “typical” teams.

3.2.3 Software Development for Research Purposes

About half a dozen teams are engaged in research work, some of it cutting edge. They are developing software to support their research rather than to support a particular GSFC mission. These teams are primarily found in Code 588. Examples are as follows.

- One team is investigating the role that agent technology can play for on-board software. This team has no actual deliverable product and no identifiable customer.
- A team is developing software for data visualization. Team members are investigating advanced data visualization technologies for mission operations. The prototypes are then demonstrated to potential users.
- Another team produces developed or integrated technologies and laboratory facilities. Their potential customers are Code 700, primary investigators, and the general public.
- A team is developing and evaluating a prototype of the Scientist’s Engineer Assistant for the Next Generation Space Telescope (NGST). They have a customer (NGST) but no real end-user, since their product will be a research prototype to demonstrate concepts.

3.2.4 Small teams

Two of the teams we interviewed are quite small, consisting of only one or two people. Both of these teams are matrixed to projects outside of ISC. Although one member of the team may be designated a “team lead”, this function is primarily administrative. The team takes its technical direction from the project to which it is matrixed.

3.3 Characterization of Teams by Team Size and Type of Software Activity

For the purposes of this study, we have characterized the teams by two criteria. First, we looked at the size of each team. Second, we looked at whether the team was developing new software or maintaining existing software (or both).

3.3.1 Size of Team

Figure 4 plots the size of the teams interviewed in Full Time Equivalents (FTEs). (NOTE: In this and subsequent figures, teams are identified by their GSFC branch codes followed by a single digit representing the order in which they were interviewed.) Two of the teams interviewed (581-1 and 585-1) do not develop or maintain any software. We have therefore deleted them both from this chart and from the discussion that follows. Two other teams (584-2 and 586-1) are very large. To better represent the relative sizes of the other teams, Figure 5 plots the same data as in Figure 4 with the exclusion of these two teams.

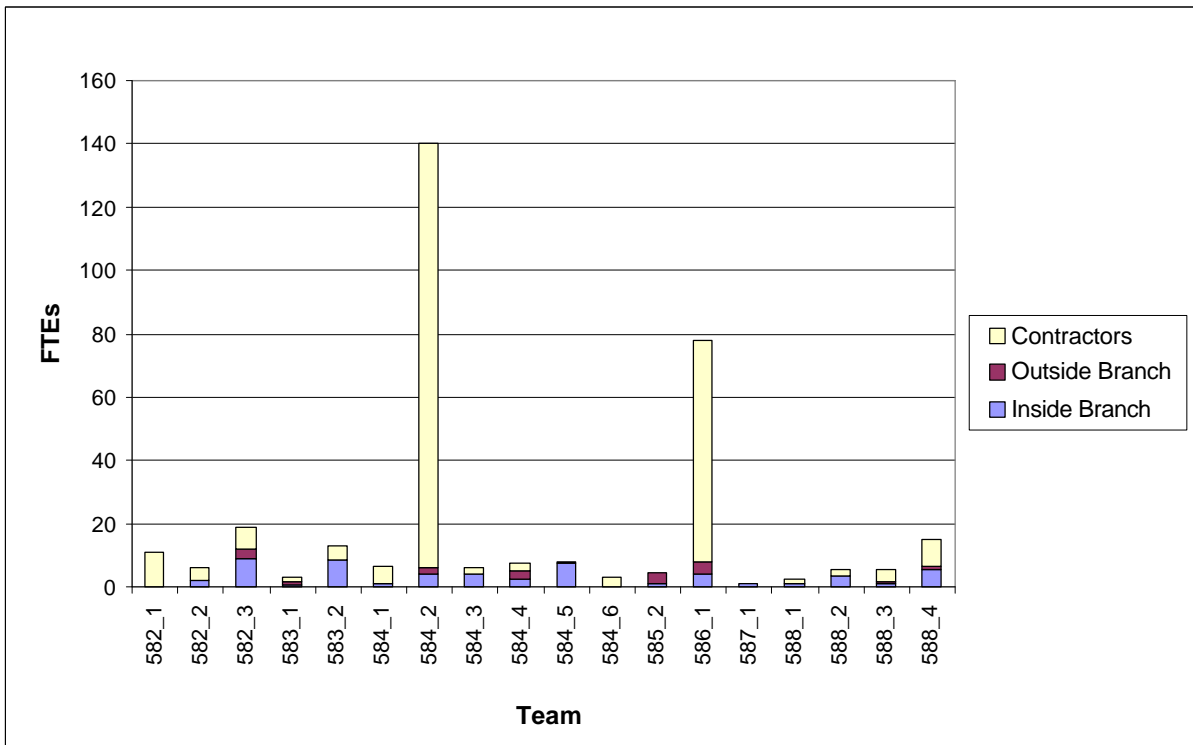


Figure 4. Breakdown of FTEs by Team

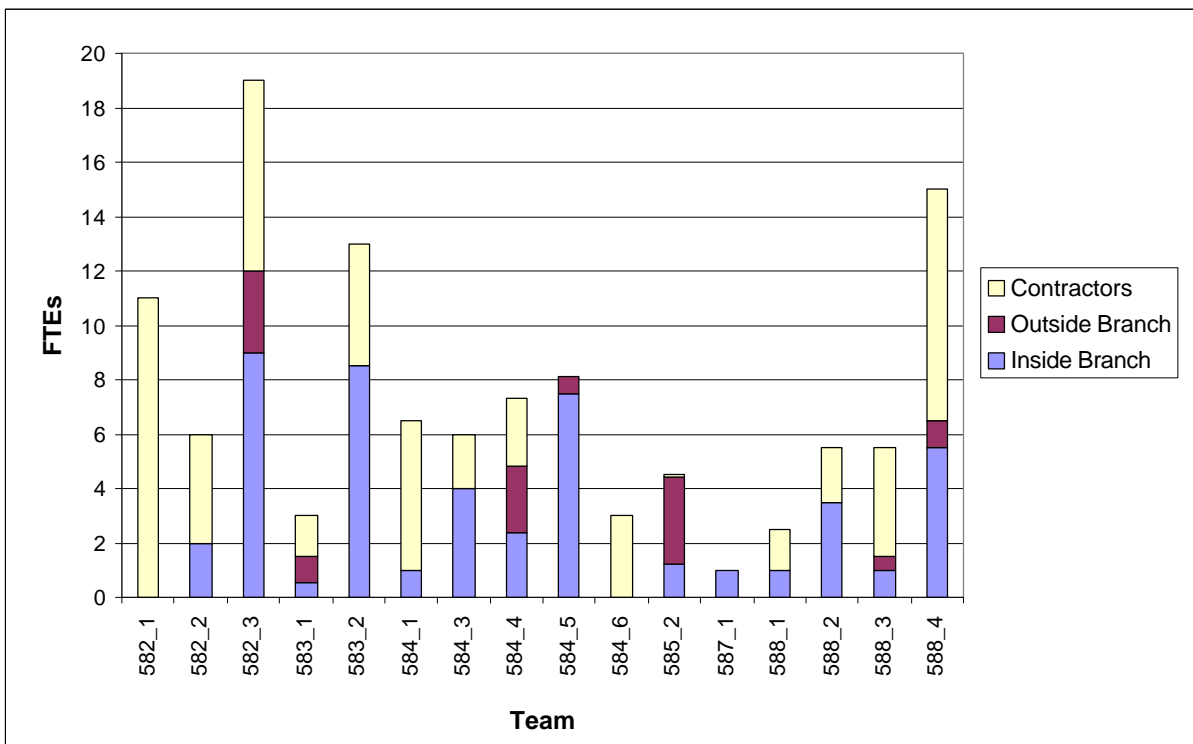


Figure 5. Breakdown of FTEs by Team Without the Two Largest Teams

In the remainder of this section we shall focus upon the 12 typical teams (as defined in Section 3.2.2) for several reasons:

- They are a relatively homogeneous category.
- They represent an important part of ISC and of its way of working.
- They represent the majority of the data sample we have gathered in this baseline.
- They represent the part of the ISC that could gain the most from process improvement initiatives.

Large project teams (e.g., 584-2 and 586-1) and teams that perform management functions have different types of problems, particularly organizational ones. Teams developing software for research purposes, by their very nature, are not subject to stringent productivity and quality constraints. Very small (one or two member) teams, especially those that are matrixed to external projects, do not typically see a project through the entire life cycle, and usually have concerns that are different from those of the “typical” teams. In most of the discussion that follows, we shall concentrate upon the 12 “typical” teams.”

Figure 6 shows the number of FTEs per typical team.

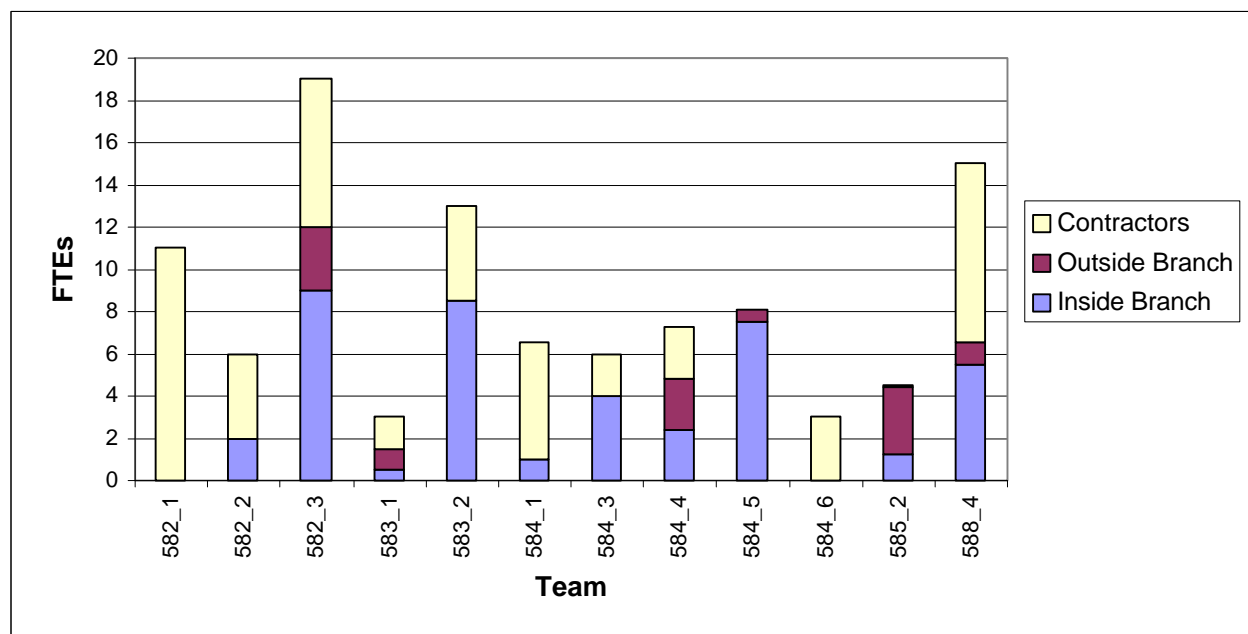


Figure 6. Breakdown of FTEs per Typical Team

3.3.2 Division of Software Effort Between Development and Maintenance Work

Figure 7 shows the percent of effort that each typical team spends on software development and software maintenance.

It is evident from this figure that the teams fall into three categories. For six of the teams, 95-100% of their effort is dedicated to new development. For two of the teams, at least 75% of their effort is allocated to maintenance. Three teams divide their work fairly equally between development and maintenance.

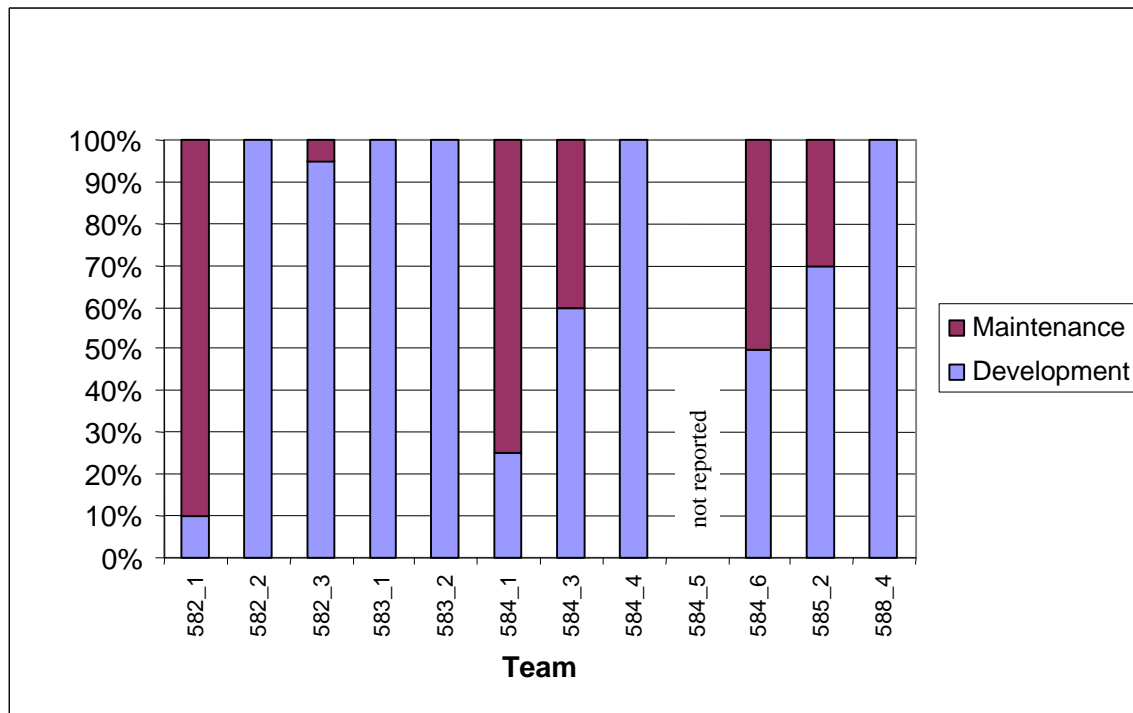


Figure 7. Distribution of Effort Between Development and Maintenance (Typical Teams)

3.3.2.1 Allocation of development effort

Figure 8 shows the distribution of development effort by activity. The teams represented are the typical teams that devote at least 30% of their entire software effort to development (cf. Figure 7).

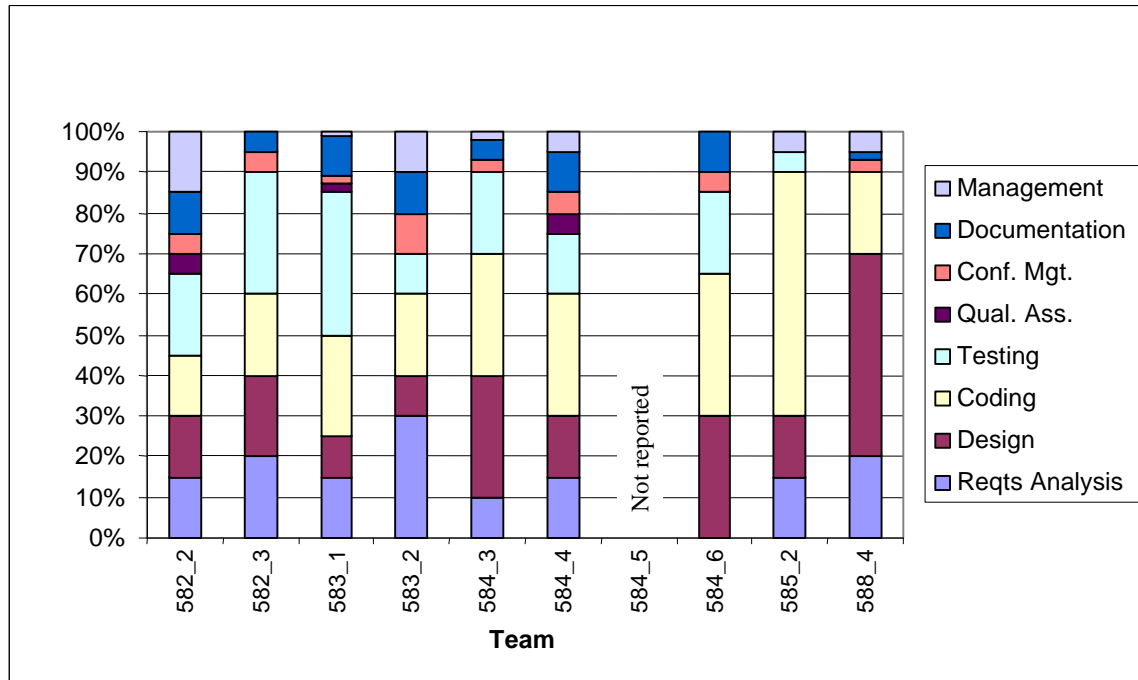


Figure 8. Distribution of Development Effort By Activity (Typical Teams)

We observe that all of the typical teams except 584-6 perform requirements analysis. (Team 584_6 indicated in their questionnaire that they are part of a larger team. Some requirements analysis is performed on the project, but team 584-6 does not participate in this phase.) Similarly, all of the teams except 588-4 report that they perform some testing. (Team 588-4 indicated in their questionnaire that they, too, are part of a larger team. Team 588-4 does not perform any formal testing.). Finally, team 585-2 performs only a limited amount of testing (5% of their development effort). This is because the software the team develops is administrative software rather than mission support software, and extensive testing was not deemed necessary. Only three of the teams claimed to spend time on quality assurance.

3.3.2.2 Allocation of maintenance effort

Figure 9 shows the distribution of maintenance effort by activity. The teams represented are the five typical teams that devote at least 30% of their entire software effort to maintenance (cf. Figure 7).

We observe that team 582-1, a flight software team, spends 60% of its maintenance time on testing. This is to be expected because of the high reliability requirements of flight software. One would not expect the same preponderance of testing hours among the teams that do not develop flight software. Figure 9 corroborates this; the remaining four teams allocate only 10-30% of their software maintenance effort to testing.

Figure 10 shows, for these same five teams, the allocation of maintenance effort among the four components of maintenance: testing/verifying, adapting, enhancing, and correcting. Once again, team 582-1 reports that a high percentage of their maintenance effort (70% in this case) is devoted to testing and verifying; this is again reasonable because of the critical nature of flight software. Similarly, team 584-1 allocates 75% of their maintenance effort to adapting. This is a much higher percentage of the overall maintenance effort than the other four teams report, and is to be expected because this team supports a variety of missions, and reports a very high percentage of software reuse.

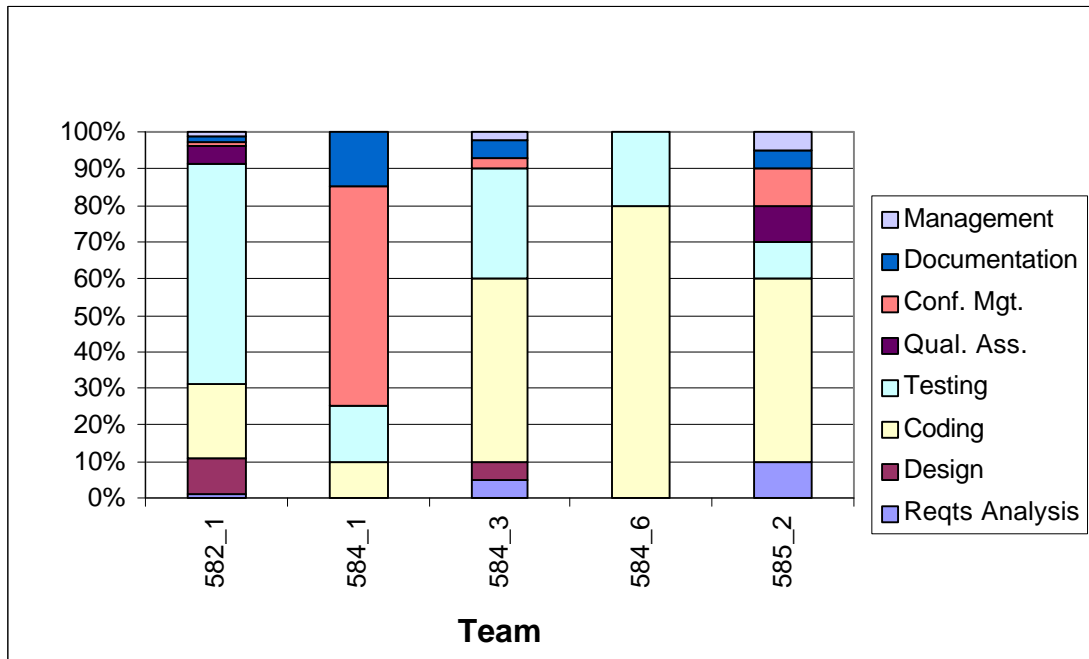


Figure 9. Distribution of Maintenance Effort by Activity (Typical Teams)

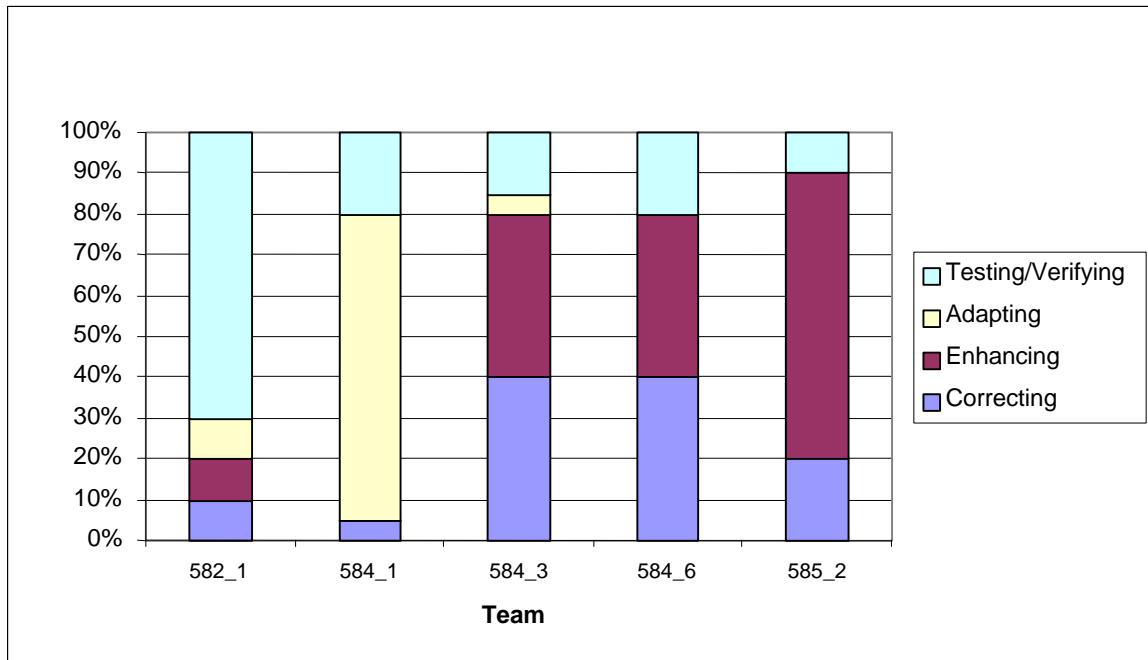


Figure 10. Allocation of Maintenance Effort by Type of Maintenance (Typical Teams)

3.4 Other Software-Related Teams Within the ISC

To verify that the FTE data presented here were representative of the ISC as a whole, particularly in relation to the number of contractors supporting the Center, we identified a list of 48 ISC teams performing software-related work. All branches of the ISC, with the exception of Codes 585 and 587, were represented in this list. We developed a short questionnaire, consisting of only eight questions, that was sent to the team leads of record. Forty-seven of those 48 teams responded to the question regarding number of FTEs. Half (24) of these 48 teams belonged to Code 588. The other half belonged to Codes 581 through 586. The data for Code 588 are therefore depicted in separate figures.

3.4.1 Sizes of Software-Related Teams

Figure 11 shows the number of civil servant and contractor FTEs, as well as the number of individual personnel, on each of the 23 software teams surveyed in Codes 581-586. Figure 12 repeats this information, leaving out the two largest teams to better show the values of the more typically sized teams. Figure 13 shows the number of civil servant and contractor FTEs on each of the 24 software teams surveyed in Code 588.

We can make several observations from these two figures. First, the teams in Codes 581-586 tend to be somewhat larger, on the average, than the teams in Code 588. The teams in Codes 581-586 average about 7 FTEs in size; the teams in Code 588 average about 2 FTEs. Second, most of the teams in all codes contain a preponderance of contractor personnel; 10 of the 23 teams in Codes 581-587 and 16 of the 24 teams in Code 588 consist of over 50% contractor personnel. Six of the 23 teams in Codes 581-586 consist only of civil servants; these are all very small teams of only one or two people. Only one of the 25 teams in Code 588 consists only of civil servants; this team, too, consists of only one person. Six of the 23 teams in Codes 581-586, and five of the 24 teams in Code 588, contain both civil servants and contractor personnel, with at least 50% civil servants.

We have added up the number of FTEs and personnel in all teams, and have determined the mean, median, and mode in each civil servant/contractor category. The resulting statistics are represented in Table 2 below. Table 2 shows that the mean number of contractor FTEs on a team is 8.9, and that there are as many teams with fewer than two contractor FTEs as there are with more than two. Additionally, the mode tells us that the most frequently occurring number of contractor FTEs is 1.

Table 2. Statistics for Civil Servants and Contractor Personnel in 48 ISC Software Teams

Category	FTE/Personnel Totals	Mean	Median	Mode
Civil Servant FTEs	102	2.2	1	1
Contractor FTEs	374	8.9	2	1
Civil Servant Personnel	146	3.2	2	1
Contractor Personnel	431	9.7	2	2

If we add up the number of civil servant and contractor personnel in all 48 ISC software teams, we arrive at a total of 476 FTEs or 577 personnel.

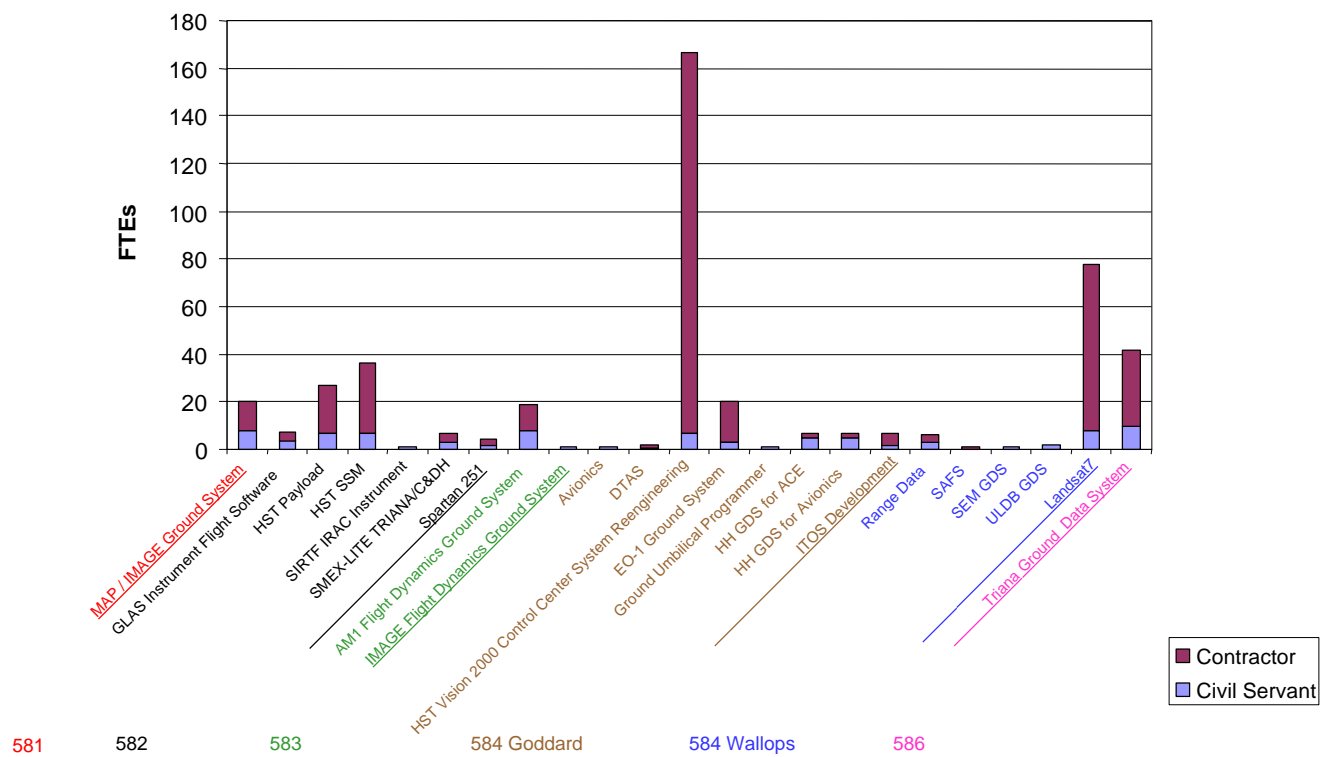


Figure 11. Civil Servant and Contractor FTEs on Software Teams (Codes 581-586)

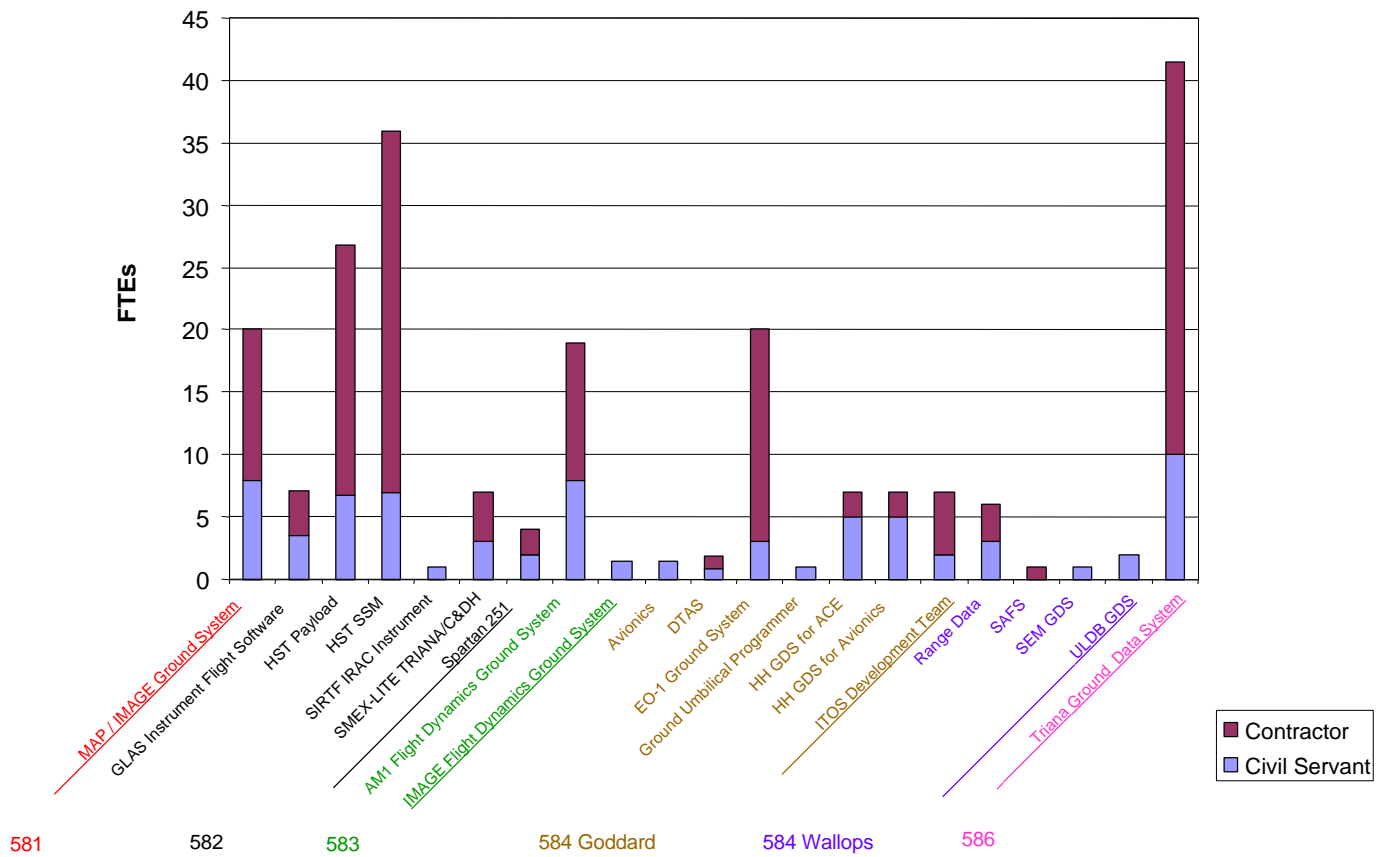


Figure 12. Civil Servant and Contractor FTEs on Software Teams (Codes 581-586), Leaving out the Two Largest Teams

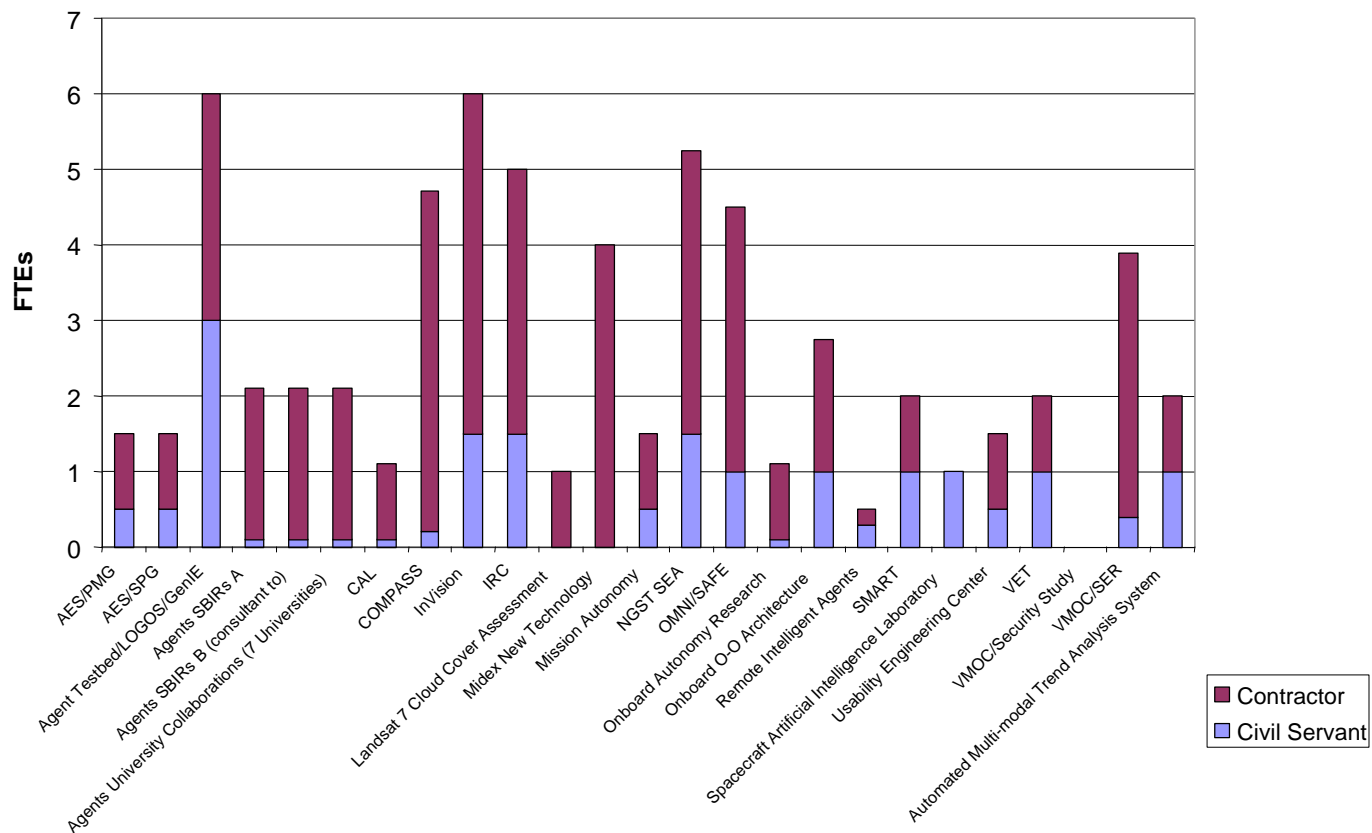


Figure 13. Civil Servant and Contractor FTEs on Software Teams (Code 588)

3.4.2 Division of Software Effort Between Development and Maintenance

Figure 14 shows how the work of the software teams in Codes 581-586 is allocated between development and maintenance. Figure 15 shows the same information for the teams in Code 588. At first glance, Figure 14 looks quite similar to Figure 7, where the same information is shown for the “typical” teams. Both figures show a preponderance of development work with a significant amount of maintenance being performed. Figure 15, however, shows that nearly all of the work in Code 588 is development; only four of the Code 588 teams are performing any software maintenance at all.

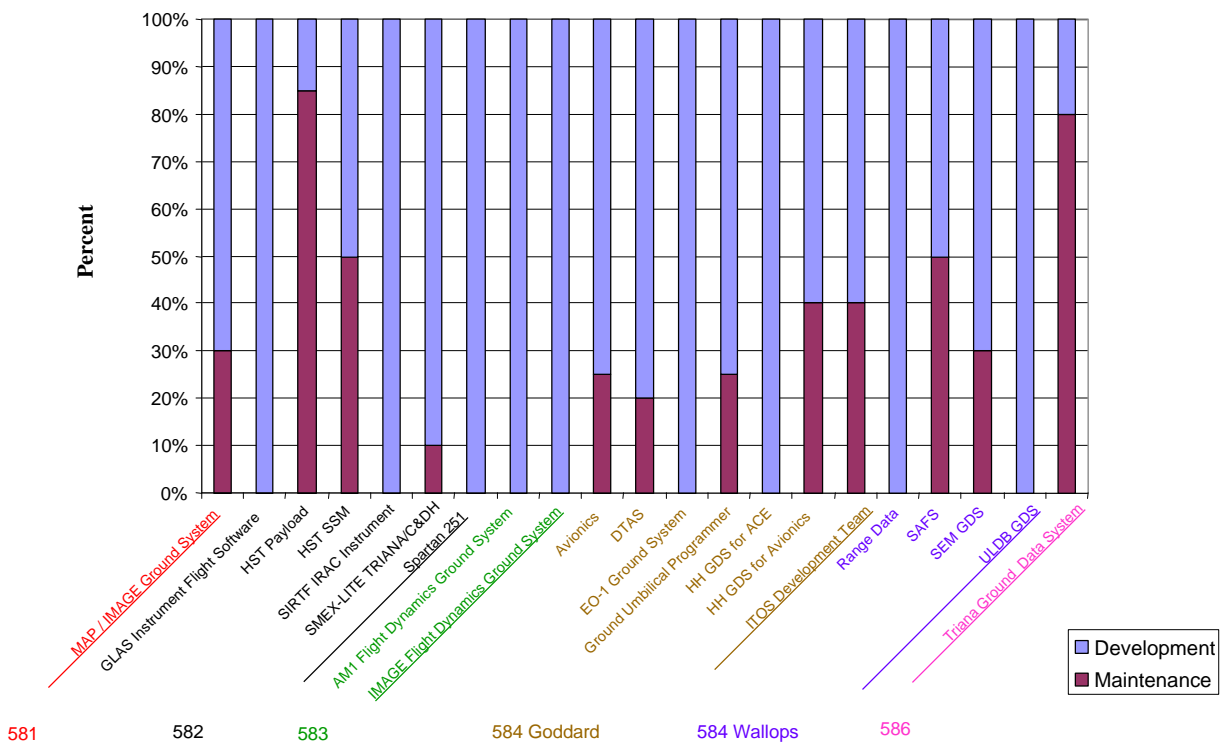


Figure 14. Distribution of Effort Between Development and Maintenance (Codes 581-586)

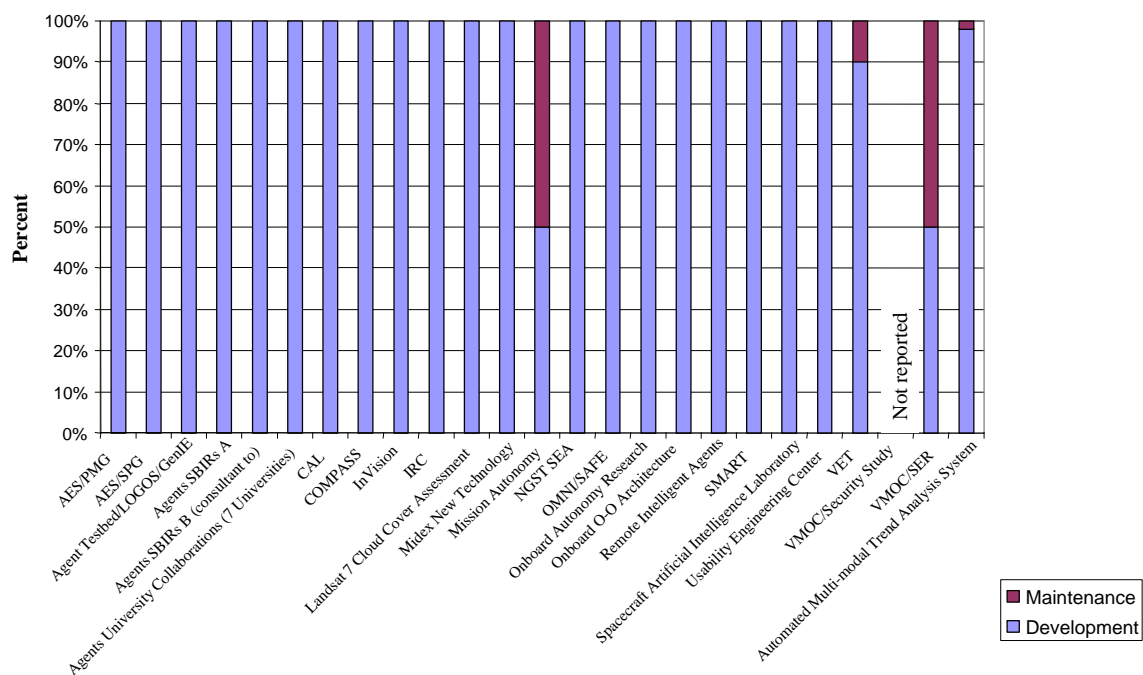


Figure 15. Distribution of Effort Between Development and Maintenance (Code 588)

4 Software Product Characteristics

4.1 Domains

At the branch level, we collected domain data by asking the branch managers to list the areas in which they worked. Their responses were not limited to domains on a predetermined list.

In the GSFC Profile (Reference 1), the NASA software was classified into six domains:

- Flight/Embedded Software
- Mission Ground Support Software
- General Support Software
- Science Analysis Software
- Research Software
- Administrative information resource management

For this study, these domains were refined to reflect the ISC environment:

- Flight Software Systems
- Mission Ground Systems (such as flight dynamics, control center, command processing)
- Information Management Support (such as Integrated Financial Management System)
- Science Processing (such as science product generation, archive, retrieval)
- Advanced Technology Initiatives (such as new techniques, prototypes)
- Other (with the teams listing what other work they are doing)

In Table 3, we characterize teams by branch code (column) and application domain (row). This table includes domain information for the 26 teams with which interviews were conducted. The numbers in the table represent the number of teams interviewed in each branch that performed work in each domain.

Table 4 describes each domain and characterizes the type of software developed.

Table 3. Number of ISC Teams Working in each Domain

	581	582	583	584	585	586	587	588
Flight Software		3		1				
Mission Ground Systems	3		2	4				
Information Mgt. Support					3			
Science Processing						1	2	
Advanced Technology								6
Other								1

Table 4. Domain Descriptions and Types of Software

Domain	Description	Type of software developed
Flight Software Systems	All software on board the satellite	Real-Time (RT) hard, embedded, communication, numerical computation
Mission Ground Systems	Flight dynamics, Mission operation and control. (Most of this software is on the ground, but part of it resides on the satellite.)	RT soft, graphical user interface (GUI), numerical computation, communication
Information Management Support	Web site development, management information systems in general (payroll, accounting, etc.)	Database, GUI, web
Science Processing	Processing and analysis of scientific data from missions	Database, GUI, numerical computation
Advanced Technology Initiatives	Prototyping, new technology usage, proof of concepts, use of cutting edge languages and techniques	Diverse
Other	Work that does not map into the above listed domains	Diverse

4.2 Amount of Operational Software

Only three branches in ISC reported collecting measures of software system size. Code 583, the Mission Application Branch, collected system size data. Codes 584 and 586, the Real-Time Software Engineering and Science Data Systems branches, reported collecting lines-of-code data. Codes 582, the Flight Software Branch, reported a sustaining engineering effort on 4 million lines-of-code for 10 operational software systems with a life-time of greater than 2-to-4 years. There is no consistent collection of system size across ISC. There are insufficient data to expand on this topic.

4.3 Development Languages

Table 5 and Figure 16 show the software development languages being used in current development and maintenance activities across ISC. Several trends are apparent. The most obvious trend is the movement away from traditional and well-established languages — particularly FORTRAN, C, and assembly— toward newer languages such as Java. This is not surprising; it is a continuation of the trend reported in the earlier GSFC Baseline Study (Reference 1). Even on projects that continue to use established languages, there is a clearly defined trend toward reduced use of these languages in favor of the newer ones. In a few cases, there is some movement away from FORTRAN or assembly toward C, or from C toward C++. These trends also reflect the overall evolution toward newer languages.

Figure 17 shows the relative use of software development languages among the five ISC domains. It is clear from this chart that choice of software development language is largely domain-specific.

Table 5. Use of Programming Languages

Branch	Team	Major Language for Development	Other Languages for Development	Major Language for Maintenance	Other Languages for Maintenance
581	581-1	N/A	N/A	N/A	N/A
582	582-1	(not reported)			
582	582-2	C (90%)	Assembly (10%)	C (90%)	Assembly (10%)
582	582-3	C (100%)	-----	C (95%)	Assembly (5%)
583	583-1	MATLAB (80%)	C (5%) C++ (5%) STK PL (5%)	FORTRAN (60%)	MATLAB (25%) C (5%) C++ (9%) STK PL (1%)
583	583-2	C, C++, Ada, Perl, shell scripts	(No percentages supplied)	C, C++, Ada, Perl, shell scripts	(No percentages supplied)
584	584-1	C (50%)	Java (25%) Misc (25%)	C (80%)	Java (10%) C++ (5%) Misc (5%)
584	584-2	C++ (45%)	Java (30%) TCL (15%) C (5%) 4GL (5%)	FORTRAN (95%)	C (5%)
584	584-3	C (50%)	C++ (25%) Visual Basic (25%)	C (80%)	C++ (10%) Visual Basic (10%)
584	584-4	C (90%)	4GL (10%)	C (100%)	-----
584	584-5	C (100%)	-----	C (100%)	-----
584	584-6	C++ (100%)	-----	C++ (95%)	C (5%)
585	585-1	N/A	N/A	N/A	N/A
585	585-2	Java/Java Script (60%)	Perl (40%)	Perl (70%)	Java (30%)
586	586-1	C (95%)	IDL (5%)	N/A	N/A
587	587-1	FORTRAN (80%)	C (15%) C++ (5%)	FORTRAN (95%)	C (5%)
588	588-1	Java (100%)	-----	N/A	N/A
588	588-2	Java (95%)	CLIPS (5%)	Java (95%)	CLIPS (5%)
588	588-3	Java (100%)	-----	Java (100%)	-----
588	588-4	Java (60%)	4GL (10%) XML (10%) Html/Java script/CFM (20%)	Html (50%)	Web objects (30%) Java-related (20%)

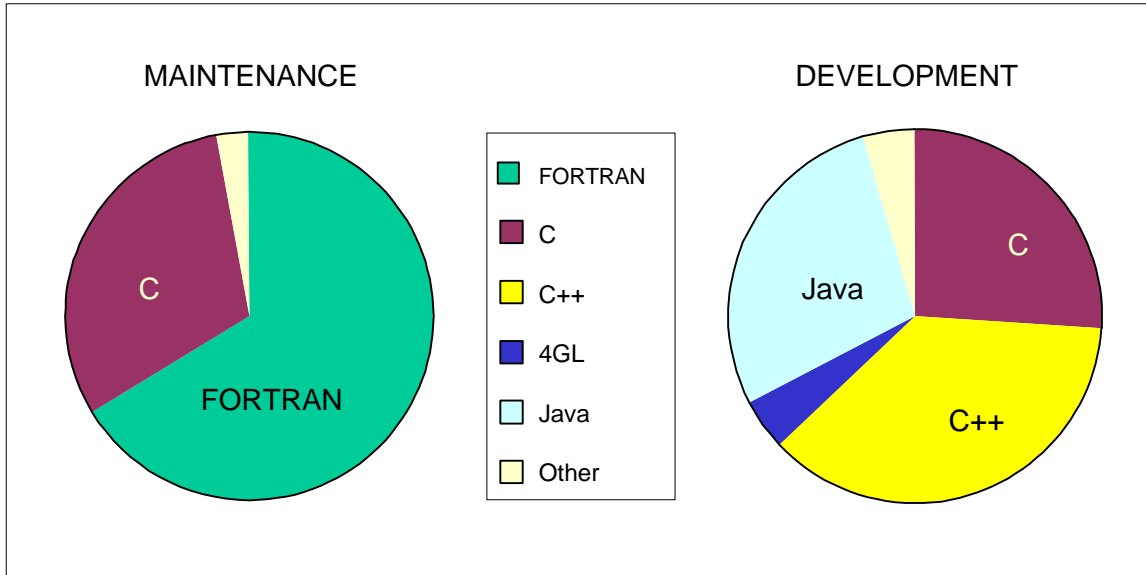


Figure 16. Programming Languages Used in Development as Compared to Maintenance

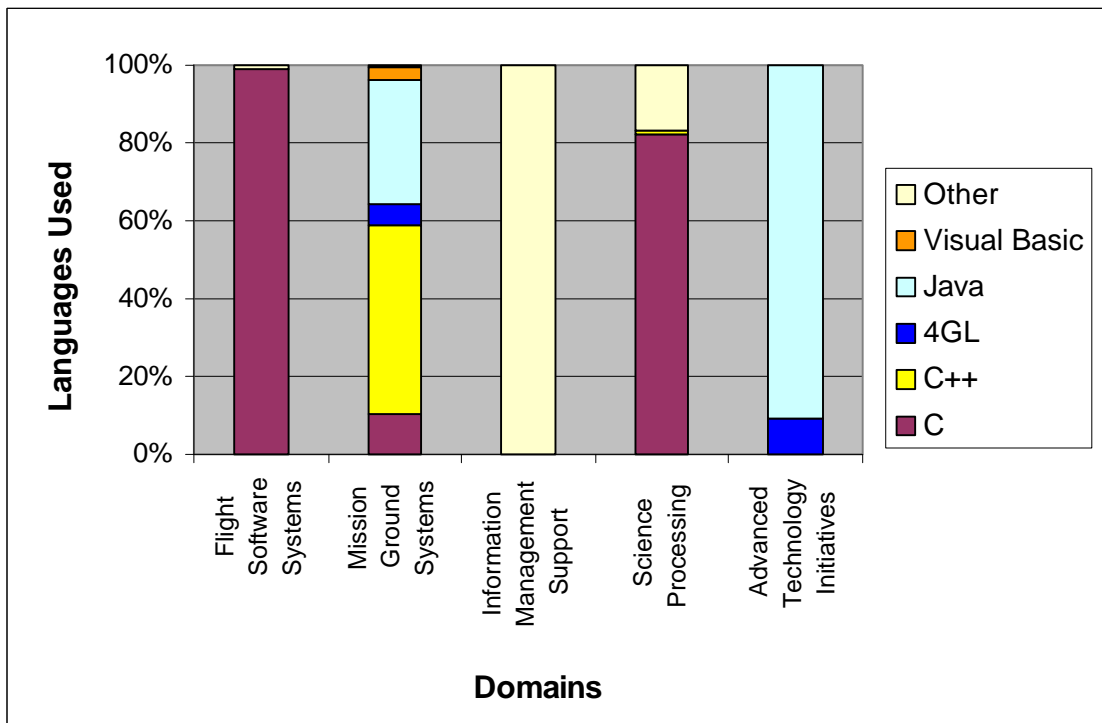


Figure 17. Development Languages Used Within Each ISC Domain

4.4 COTS Implementation

In the questionnaire, use of COTS products was divided into three categories, which were defined as follows:

- embedded - “COTS products [used] as components of deliverable systems”;
- delivered - “products [used] to support software development and maintenance that must be delivered with a system”, and
- non-delivered - “products [used] to support software development and maintenance that are NOT delivered with a system.”

The definition of COTS is broad enough to cover any software developed by a third party, but in this study we focus on those COTS products that are used immediately in the product as a part of it (embedded COTS). These products can be used as libraries or stand-alone applications. Languages and operating systems may be considered “off-the-shelf” if the developed product extensively uses their libraries. We call embedded COTS products that are not operating systems or languages “real” COTS. The typical “real” COTS products used in ISC are DBMSs (e.g., ORACLE) and GUIs (e.g., Motif). An expert system engine, such as Advisor/J, is another example of an embedded COTS product.

Table 6 lists the COTS products used within each ISC branch and categorizes them as **Embedded**, **Delivered**, or **Non-delivered**. The delivered and non-delivered COTS include languages (MATLAB, J Builder, Java Script), compilers, linkers, and operating systems (SCO UNIX, Linux, Vxworks). Some non-delivered COTS are development frameworks or tools, such as Metaware High C Compiler or Visual Café.

We can conclude that the ISC teams make significant use of COTS products. Although they mostly represent horizontal reuse (traditional OS and language libraries, GUIs, DBMSs and other general-purpose software), domain-oriented software products, such as ENVI (visualization of satellite data) are used as well. Another conclusion is that embedded COTS products are used primarily by “typical” teams; they are not used by small teams and are used very little by research teams.

Table 6. Use of Embedded, Delivered, and Non-Delivered COTS Products

COTS CATEGORY	COTS PRODUCTS	VENDOR	581	582	583	584	584	585	586	587	588
						GSFC	Wal.				
Aerospace											
	Autocon	AI Solutions, Inc.			E&D						
	Free Flyer	AI Solutions, Inc.			E&D						
	GREAS		E								
	Satellite Tool Kit (STK)	Analytical Graphics	E		E&D						
Scientific/Engineering											
	Altair	Altair Engineering, Inc.									E
	CADRE	CADRE Analytic							N		
	ENVI	Better Solutions Consulting							E		
	MATLAB	Mathworks			E&D					E	
	Sat Track	Bester Tracking Systems					E&D				
Chart/Graph											
	XRTGraph	KL Group				E					
CASE	(in general)	-----	E								
	System Architect					D					
CM	(in general)	-----									D
	GNU CVS	Free Software Foundation				D					
	PVCS			N					N		
	RCS	Component Software			N						
	SCCS				N						
	TruChange					D					
	Visual Source Safe	Microsoft									N
DBMS	(in general)	-----		D & N					E		
	MS SQL	Microsoft									E
	ORACLE	Oracle Corp.			E	E			E		
	RDBMS (in general)	-----									E
Development Framework/ Environment											
	Delphi, incl. J. Builder	Inprise (Borland)				D					D
	Motif					E	E&D				
	Visual Café	Symantec									N
Development	(development tools in ge	-----									N
	various development too	Rogue Wave Software				N					
	compilers (in general)	-----		D							D
	Metaware High C compiler			N							
	linkers (in general)	-----		D							
	Phar Lap Linker Locator			N							
	debuggers (in general)	-----		D							
	gdb (GNU source-level debugger)	Free Software Foundation				D					
	make utilities (in general)	-----		D							
	MS Visual C++	Microsoft				D					
	Borland C++	Inprise (Borland)				D					
	C++ libraries										E
	TBD C programming too	-----					N				
	Interface Development Language									E	
	Perl	Free Software Foundation				D					

Table 6 (cont.). Use of Embedded, Delivered, and Non-Delivered COTS Products

COTS CATEGORY	COTS PRODUCTS	VENDOR	581	582	583	584	584	585	586	587	588
						GSFC	Wal.				
	Code Analysis (static and dynamic) tools	-----									D
	test software (in general)	-----				N					
	Purify	Productivity Through Software			N				N		
Internet/Intranet/Web											
	web page tools (general)	-----									N
	web clients	-----							E		
	web objects	-----									E
	Advisor/J	Blaze Software									E
	AVS	AVS Computers								E	
	Cold Fusion	Allaire									E
	Data Vista charting widgets										E
	JDK	Sun			N						
	Lotus E-Suite	Lotus									E
	Lotus Notes	Lotus									E
	OrbixWeb (CORBA)	Iona Technologies									E
File Transfer											
	AMASS	EMASS, Inc.				E					
	FastCopy	SoftLink					D				
Systems Monitoring											
	PATROL	BMS Software				E					
Operating System											
	in general	-----	E	E							
	DOS						D				
	Epoch 2000		E								
	LabVIEW	National Instruments					E&D				
	Linux	Linus Torvalds					E&D				
	SGI workshop bundle	Silicon Graphics							D		
	Unix, including SCO Unix					D	D				
	VRTX			E							
	VxWorks	Wind River Systems		E			E&D				
	Windows NT						D				
Other											
	commercial C2P systems		E								
	Other Vision 2000 COTS	-----	E			E					
	CD-ROM generators	-----							E		
	Formula 1							E			
	GOTS-Assist				E&D						
	ICO		E								
	Message Passing Interface									N	
	Net charts							E			
	Parallel Virtual Machine (free through netlib)									N	
	RGS				N						
	Structure Builder										N
	TL Executive						D				

4.5 Operational Lifetime of Software

We also categorized the software systems in the various branches according to their expected operational lifetimes. Code 585 is supporting software systems with operational lifetimes of less than 2 years. ISC Codes 582, 583, and 586 reported having software of more than 2 years operational lifetime. Codes 582 and 585 reported having software systems with expected operational lifetimes of greater than 2 to 4 years. Code 586 reported having six systems with operational lifetimes of greater than 7 years.

4.6 Software Error Characteristics

Software errors were divided into two groups: external (introduced by persons or factors outside the team) and internal (introduced by team members). The following causes of errors were considered as external: changing requirements, missing requirements, and environment problems. The following causes of errors were considered as internal: misinterpreted requirements, design errors, and coding errors. Interfaces are considered as causing both external and internal errors. For some teams, errors stemmed from predominately internal causes. For other teams, external errors dominated. Among fairly small development teams (size from 2.5 to 8 FTEs), 70% have predominately external errors. Both small maintenance teams (one or two FTEs) have internal errors dominating.

Out of three large development teams (size from 11 to 20 FTEs), one team has external errors dominating, and two teams have internal errors dominating. The large maintenance team has more problems with internal errors. Among the two very large teams (over 50 FTEs), the team in Code 584 has more problems with external errors, while the team in Code 586 has more problems with internally caused errors.

4.7 Requirements Stability

Over half of the teams that completed questionnaires answered that their requirements were either “fairly stable” or “very stable” with only a little over 25% answering “unstable.” The remaining teams did not respond to this question. Embedded Software teams and Mission Ground Systems had the most stable requirements. This result is expected because these teams are developing or maintaining the most safety-critical software. Information Management Support teams, Science Processing teams, and Advanced Technology teams have the least stable requirements. This result is also expected because these types of software are driven by the constantly changing needs of other users.

5 Software Process Characteristics

An organization with a mature software development and maintenance process is one in which key processes are a routine, ingrained part of its culture. These processes include disciplined requirements management, quality assurance, measurement of quality and process, and configuration management. A mature organization is characterized by a uniform core process that is tailored in approved ways for each specific application (Reference 7).

This chapter focuses on the current software process characteristics across the ISC community.

5.1 Software Processes and Standards

We explored the presence of advocated software processes and standards and the extent to which these processes and standards were known, helpful, and used. We found that the use of software processes and standards varied considerably from branch to branch, and even varied significantly among different projects within the same branch. Three branches reported more extensive use of software standards than of processes; one branch reported the reverse.

In the questionnaire, we defined software process as “the phases, activities, and products by which the software is defined, developed, documented, and delivered; such a process would include policies and standards, formal and informal reviews, and collection, analysis, and use of metrics.” We defined software standard as a specification of the format and content of a software product or document.

Table 7 presents the data submitted by the teams that performed software development and/or maintenance. For each such team, the table shows how much use (minimal, some, or extensive) the team made of both processes and standards, and how helpful (minimally, somewhat, or very helpful) they were felt to be.

5.1.1 Software Development Processes and Standards

Team leads in all branches except 581 reported that they do some software development work. Use of processes and standards was often quite high, and was sometimes reported at 100%. Use of processes and use of standards were not always the same. Team leads in three branches reported higher use of standards, while the team leads in one branch reported higher use of processes. Most team leads reported some use of both processes and standards; a few teams reported “extensive use”, and a few team leads reported “minimal use.” Similarly, most team leads reported that the processes and/or standards were “somewhat helpful” and a gratifying number of teams reported that they were “very helpful.” A single team lead, who reported no use of processes, reported that the standards used were “minimally helpful.”

Only one branch reported that they used software development standards that had been prepared and disseminated at the branch level. Most processes and standards are selected and applied at the project level.

5.1.2 Maintenance Processes and Standards

Team leads in Codes 582, 584, and 585 reported that they performed some software maintenance work. As with the software development work, reported use of processes and standards was generally quite high, frequently 100%. The levels of process and standards use were usually equal, although the team leads performing maintenance work in one branch reported higher use of processes. Most team leads reported some use of both processes and standards. A few teams reported “extensive use”, and one team lead performing maintenance reported no use of standards. About half of the team leads performing maintenance reported that the processes and/or standards were “somewhat helpful;” the other half reported that they were “very helpful.”

It was clear from our discussions that the selection of processes and/or standards to be followed for software maintenance, as for software development, varied significantly from project to project. None of the branches performing software maintenance work reported that they used standards that were prepared and disseminated at the branch level. In general, as with software development, it is at the project level that software processes and standards are selected and applied.

Table 7. Reported Use and Helpfulness of Software Processes and Standards

Team Identifier	Team Performs Development	Team Performs Maintenance	Reported Use of Processes	Reported Helpfulness of Processes	Reported Use of Standards	Reported Helpfulness of Standards
582-1	X	X	Extensive use	Very helpful	-----	-----
582-2	X		Some use	Somewhat helpful	Some use	Very helpful
582-3	X	X	Some use	Very helpful	Extensive use	Somewhat helpful
583-1	X		Some use	Somewhat helpful	Some use	Somewhat helpful
583-2	X		-----	-----	Extensive use	Very helpful
584-1	X	X	Some use	Somewhat helpful	Some use	Somewhat helpful
584-2	X	X	Extensive use	Very helpful	Extensive use	Very helpful
584-3	X	X	Extensive use	Very helpful	Some use	Somewhat helpful
584-4	X	X	Some use	Very helpful	Some use	Somewhat helpful
584-5	X	X	Some use	Very helpful	Some use	Somewhat helpful
584-6	X	X	Extensive use	Very helpful	Extensive use	Very helpful
585-2	X	X	Some use	Somewhat helpful	Some use	Somewhat helpful
586-1	X		Extensive use	Very helpful	Extensive use	Very helpful
587-1	X		-----	-----	Minimal use	Minimally helpful
588-1	X		-----	-----	Some use	Somewhat helpful
588-3	X		-----	-----	Extensive use	Very helpful
588-4	X		Some use	Somewhat helpful	Extensive use	Very helpful

5.1.3 Documentation Standards

Documentation serves as a primary means of communication among all of the parties concerned with a software development activity. Key documents, however large or small, convey information such as planned budgets and steps, user requirements, and design specifications. Necessarily, the larger the project, the more difficult the communication and coordination task and the more critical the quality of the documentation (i.e., clear, well written, and comprehensive).

We explored the role of documentation in the types of software development and maintenance performed in ISC. In the team-level questionnaire we asked, “Which of the following key project documents does your team generally produce?” and listed seven documents: User’s Guide, Requirements Specification, Design Document, Test Plan, Quality Assurance Plan, Project Plan, and Configuration Management Plan. The team leads’ responses are charted in Figure 18.

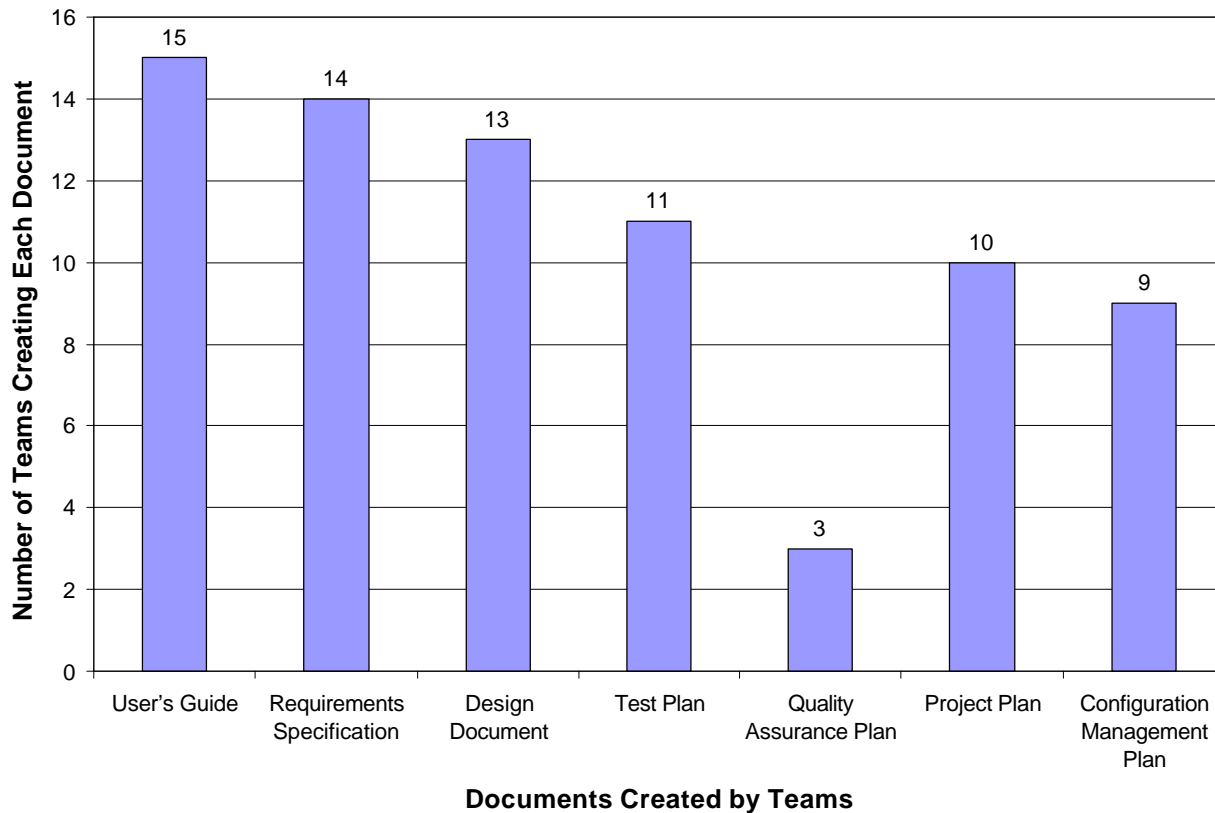


Figure 18. Key Documents Produced

Ten of the team leads interviewed reported that they used Project Plans. Of these ten team leads, six reported that the project plan was kept current and followed; the other four reported that the project plan was followed but not maintained. Only three of the team leads reported that they produced Quality Assurance Plans.

A few team leads reported that they used the following documents, which were not included on the pick-list:

- Operations Concept Document
- Program Status Reviews
- Software Maintenance Manual
- Programmer's Reference Guide/Manual
- Risk Assessment Plan
- Release Implementation Plan
- Revised Interface Control Documents
- Transition Plan

5.2 Project Management Practices

5.2.1 Management experience

The team leads were categorized by whether they were in-house (civil servants) or contractor personnel. They were also asked to indicate how many years of team or project leadership experience they had. We then grouped them into four categories: 0-2 years, 3-5 years, 5-10 years, and over 10 years of management experience, as shown in Table 8 below. "CS" indicates team leads who are civil servants; "C" indicates team leads who are contractors. One team lead did not respond to this question.

Table 8. Leadership Experience of Team Leads, by ISC Branch

Code	0-2 Years Leadership Experience	3-5 Years Leadership Experience	5-10 Years Leadership Experience	Over 10 Years Leadership Experience
581				CS
582		C	C	
583	CS			CS
584 (GSFC)	C	CS	CS	CS
584 (Wallops)	CS	CS		
585				CS CS
586			CS	
587				CS
588	CS CS	CS		CS

5.2.2 Matrixing

Of the ISC branches answering this question, all reported some matrixing of staff to other ISC branches or to organizations outside the ISC. Codes 582, 584 (Wallops), and 587 reported that 95%, 80% and 100% of their staff, respectively, were matrixed to organizations outside the ISC. Code 582, for example, delivers on-board flight software to the Projects. Code 586, whose mission is to support the Earth and Space Sciences efforts at GSFC, reported that 75% of its staff were matrixed to outside organizations. Codes 583, 584 (Goddard), 585, and 588 reported 25%, 50%, 20%, and 30% matrixing of staff outside the ISC, respectively.

5.3 Software Engineering Practices

5.3.1 Development Methodologies

Both the branch managers and the sample of team leads were asked to report the level of awareness, training, and usage of the following software development methodologies. Each branch manager reported for the branch as a whole. Each team lead reported for just his or her own team. They are:

- Formal Methods
- Cleanroom Techniques
- Inspection/Walkthroughs
- CASE Tools
- Prototyping
- COTS Integration
- Structured Analysis
- Object-oriented Methods
- Information Hiding
- Reliability Modeling (branch management only)
- Defect Causal Analysis (branch management only)

For each methodology, respondents had a choice of responding “minimal,” “some,” or “much” for the three categories of awareness, training, and usage.

A comparison of the team and manager responses was conducted as follows. Numeric values were assigned to the responses in the following manner: minimal = 1, some = 2, much = 3. For each question (e.g., “What is the {branch’s | team’s} awareness of prototyping?”) the team response was paired with the response of the branch

manager to the same question. After eliminating questions where either the branch manager or all of the manager's sample teams failed to reply, there were 278 data set pairs remaining.

In the ideal case, we would expect each data pair to be from the set $\{(1,1), (2,2), (3,3)\}$, where the branch manager's response is represented by the first number in each pair, and the team lead's response is represented by the second number. Such an ideal outcome would demonstrate that the branch manager's judgement about the branch's awareness, training, or usage of a particular methodology was exactly matched by the judgements of the sample team leads of that branch. This close agreement between the branch manager and the sample team leads would strongly suggest that the reported values were correct.

The actual data pairs, graphed in Figure 19, fall somewhat short of this ideal result. All nine data pair possibilities $\{(1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3)\}$ occurred. The frequency of each of the nine data pair possibilities is denoted numerically by an integer and visually by the size of the square at those coordinates of the graph. Three of the four highest frequencies occur at the four desired pairs: $(1,1)$, $(2,2)$, and $(3,3)$. Were it not for the fact that the highest frequency, 58, occurs at $(2,1)$, this distribution of frequencies would be close to ideal. Overall, the branch managers agreed with the team leads in about 40% of their responses.

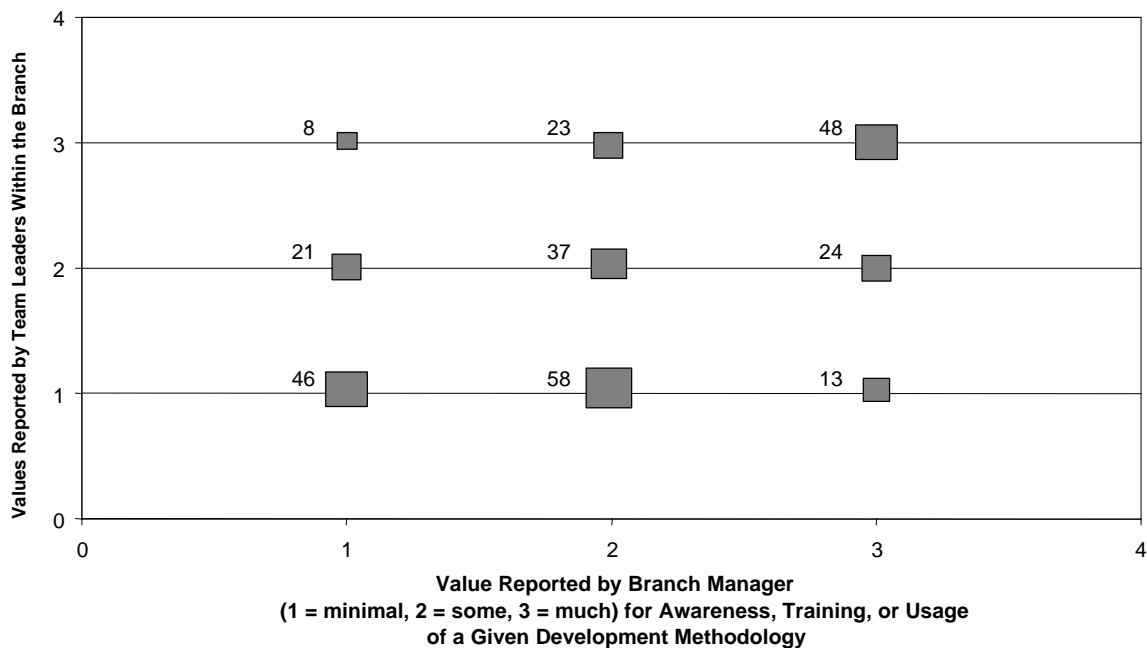


Figure 19. Comparison of Team Responses (Averaged by Branch) Versus Branch Manager's Responses for Awareness, Training and Usage of Nine Development Methodologies

Looking in more detail at the team lead responses, we observe the following about the acquaintance of these particular teams with these development methodologies.

- There was significant awareness of formal methods, as evidenced by 11 of the 18 teams reporting their awareness as “some” or “much.” Usage was more scant, however, with only team 582-1 reporting it as “much” and five teams in Codes 584, 586, and 588 reporting it as “some.” The level of training was about the same as that of usage.
- Nearly all teams reported “minimal” awareness, training, and usage of the Cleanroom methodology.
- Although these teams knew little about Cleanroom, they were well acquainted with one major feature of Cleanroom, the emphasis on inspections. Nearly all of the teams rated their awareness of inspections and walkthroughs as “much,” and 11 of them rated their usage as “much” or “some.” Only in training were they weak, with only three teams reporting training as “much” or “some.”
- Only five teams evidenced significant involvement with CASE tools. All five reported “some” or “much” awareness and usage, and four of the five reported “some” or “much” for training as well.
- Awareness of prototyping was strong across this sample set. Training was again “minimal” in most cases. Usage of prototyping was reported as “much” by seven teams and as “some” by four more teams.
- Usage of COTS integration was strong, with six teams reporting it as “much” and six more as “some.” Awareness was slightly higher, but again training was much lower. Only five teams reported “some” training and one reported “much” training.
- Awareness of structured analysis was moderate, with 11 teams reporting “some” or “much.” Usage was less evident, with only two teams reporting “much” and three more reporting “some.” Training slightly outdistanced usage: seven teams reported “some” or “much.”
- The strongest showing was clearly in object-oriented methods (O-O). Fifteen teams reported “some” or “much” awareness of O-O. Six of these teams reported “much” training, and seven more reported “some” training. Seven of these teams reported “much” O-O usage, and four more reported “some” O-O usage.
- Although information hiding is an important aspect of object-oriented programming, these teams evidenced much less involvement with information hiding than with object-oriented methods. Only nine teams reported “some” or “much” awareness of information hiding. Only three teams had received “some” or “much” training in it. Only six teams were making “some” or “much” usage of information hiding.

5.3.2 Testing Methods

We studied the range of testing methods used among the projects we surveyed. It should be noted that we did not use a pick-list here, because we wanted to explore the range of testing methods used, rather than constrain the respondents to select from a rather limited (and possibly out-of-date) list. This choice probably introduced some inaccuracy into our data. Many of the team leads asked “What do you want here?”, and the answer that the interviewer provided sometimes influenced the response. When similar questions were asked and pick-lists were provided, the results were sometimes quite different (cf. Section 5.3.6).

The testing methods being used, and the teams that report using each method, are shown in Table 9. Note that only data from teams that perform formal testing are reported in this table.

Table 9. Use of Testing Methods in ISC Teams

Testing Method	582-1	582-2	582-3	583-1	583-2	584-2	584-3	584-4	584-5	585-2	586-1	588-1	588-2	Total
Software simulator		X				X								2
Unit/module testing			X				X		X	X		X	X	6
Build testing			X											1
Process testing													X	1
Integration & testing													X	1
Independent testing									X	X				2
Beta testing									X	X				2
Functional testing	X			X										2
End-to-end testing				X			X							2
White box testing											X			1
Black box testing					X						X			2
Regression testing	X				X	X	X							4
Acceptance testing			X											1
Performance testing					X									1
Stress testing							X							1
Requirements traceability matrix								X	X					2
Automated script testing						X								1
Total	2	1	3	2	3	3	4	1	4	3	2	1	3	32

We should note that the fill-in-the-blank approach to this question resulted in a proliferation of testing methods reported. Had we used a pick-list, we believe that there would have been a much higher degree of clustering. We suggest, though, that the list of terms we collected would make a good basis for a pick-list to be used for future surveys.

Team leads were asked three follow-up questions on testing; their responses were as follows:

QUESTION	YES	NO	“NOT SURE”	“IT DEPENDS”
Is test data archived?	11	6		1
Are forms used to record test results?	9	9		
Is there training for testing?	3	14	1	

It is apparent from our small sample of the ISC software teams that archiving of data is well established within ISC; well over half the teams surveyed report that they archive test data. Only about half the teams, however, report that they use forms to record test results. Training for testing is offered only rarely.

5.3.3 Software IV&V

Verification and Validation is a system engineering process employing a variety of software engineering methods, techniques, and tools for evaluating the correctness and quality of a software product throughout its life cycle. Verification is the process of determining whether or not the products of a given phase of the software development cycle fulfill the established requirements. Validation evaluates software at the end of the development lifecycle to ensure that the product not only complies with the specific criteria set forth by the customer, but performs as expected.

Classically, independent Verification and Validation (IV&V) is performed by an organization that is technically, managerially, and financially independent of the development organization. A semi-independent IV&V approach that is sometimes employed utilizes an IV&V agent from the development organization who did not participate in the development effort.

Our findings indicate that software IV&V is rarely used within ISC. Only one team lead reported the use of independent testing (validation), and none of the team leads volunteered that they employed independent verification.

5.3.4 Development Tools

We tabulated the types of tools most often used in current software development and support efforts across the ISC. Without defining “development tools,” we simply asked the team leads to list the development tools that they used. We found that ISC projects routinely apply the following types of tools:

- Debuggers (11 teams)
- CM aids (8 teams)
- Documentation tools (8 teams)
- Design/graphics (7 teams)
- Test data generators (6 teams)
- Traceability (5 teams)
- Requirements analysis (5 teams)

A few teams also reported use of the following, some of which are actually problem areas rather than specific development tools:

- Test coverage (2 teams)
- CSP (1 team)
- Java IDE – J Builder (1 team)
- VML design (1 team)
- Complexity measuring (1 team)
- Memory leaks (1 team)

Most of the teams seem to be applying such tools in a standalone mode; we did not find significant use yet of integrated tool environments. Reported usage for these various types of tools is depicted in Figure 20.

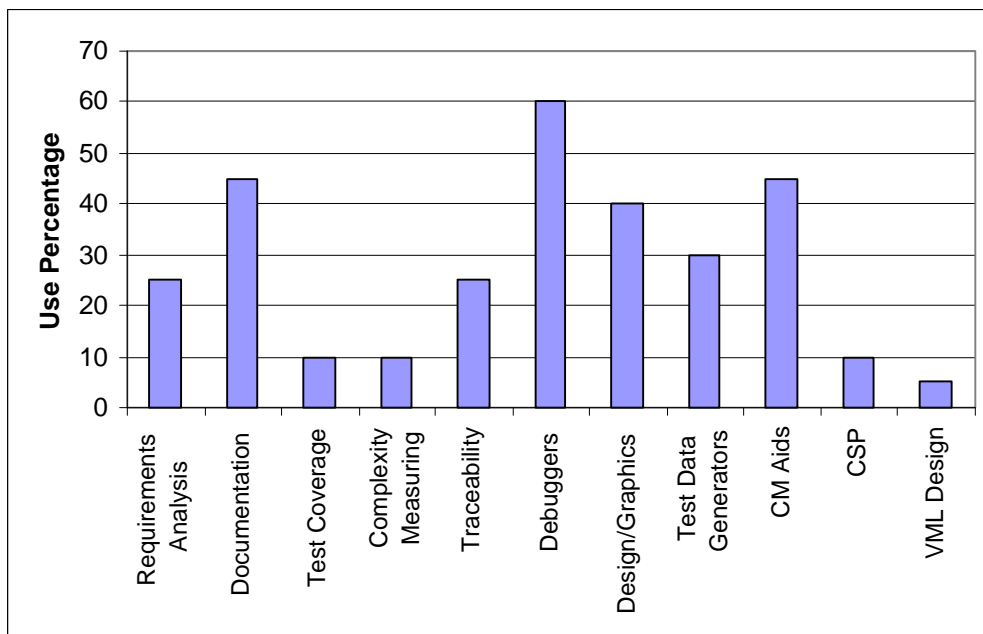


Figure 20. Use of Routinely Applied Tools

The most extensive use of tools was reported by the teams in Codes 584, 586, and 588. The teams in Codes 582, 585, and 587 reported that they used two or three tools each. No tool use was reported by the teams we interviewed in Codes 581 or 583. It should be noted that our sample was limited and possibly skewed; see Section 2.1 for a discussion of sample set coverage.

In terms of the different types of teams we have identified in Section 3.2, the most extensive use of tools was reported by the teams developing software, whether for operational use or for advanced technology. Slight use of tools was reported by the small teams, and none by the teams performing management functions.

5.3.5 Software Reuse

The level of software reuse reported by each team is shown in Figure 21. Among most of the teams from whom the SEL received questionnaires, the percentage of reused software is 30% or less. There are only four exceptions. The team that performs integration testing of flight software and command and data handling software for the Small Explorer missions reported 95% code reuse. The team developing the control center for the Ultra Long Duration Balloon reported 70% code reuse. A one-person team in Code 587 that performs computation intensive scientific modeling for both Earth and space science also reported 70% code reuse. A very large team in Code 586 reported 45-50% reuse in one of their software systems, but only 5-10% reuse in their other systems. Although three of the five teams showing zero reuse perform software-related work, they do not actually develop code.

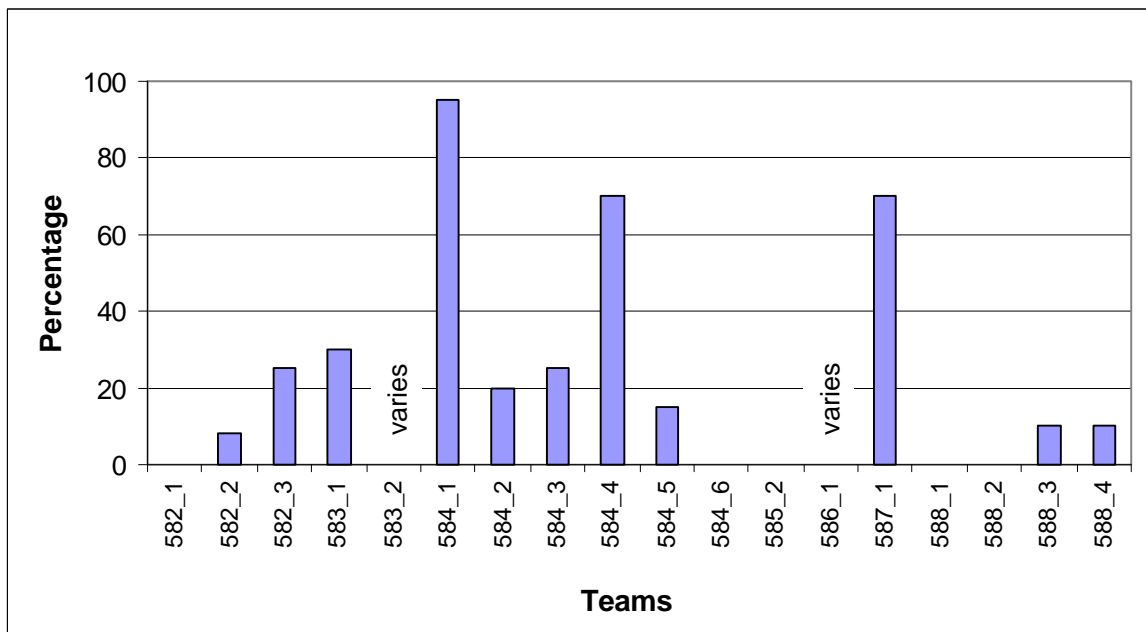


Figure 21. Percent of Reused Code

5.3.6 Maintenance Practices

The goal of software maintenance is to preserve the integrity of the software product while enhancing its functionality over time. A number of practices that protect that integrity have become broadly accepted in the software profession over the past decades. These practices include the following:

- Using change request forms to provide a standardized, documented account of each proposed change;
- Performing formal impact assessment of each proposed change to be sure that the costs, schedule impact, and associated risks are understood before deciding whether to implement or postpone the change;
- Managing and communicating change through the mechanism of a change control board;
- Conducting regression tests to verify that the change did not affect the product in an unexpected manner;
- Keeping the product's documentation current with the software;
- Collecting and analyzing software measurements (metrics) to monitor the quality and status of the software product and process and to identify elements of the process that could be improved.

We surveyed the use of these maintenance techniques among the teams we interviewed. The results are shown in Figure 22. The numbers indicate what percentage of the time the teams reported the use of each technique. We should note that we used a pick-list for this portion of the survey; this may explain why the regression testing data are not consistent with those presented in Section 5.3.2, where the questions were "fill-in-the-blank."

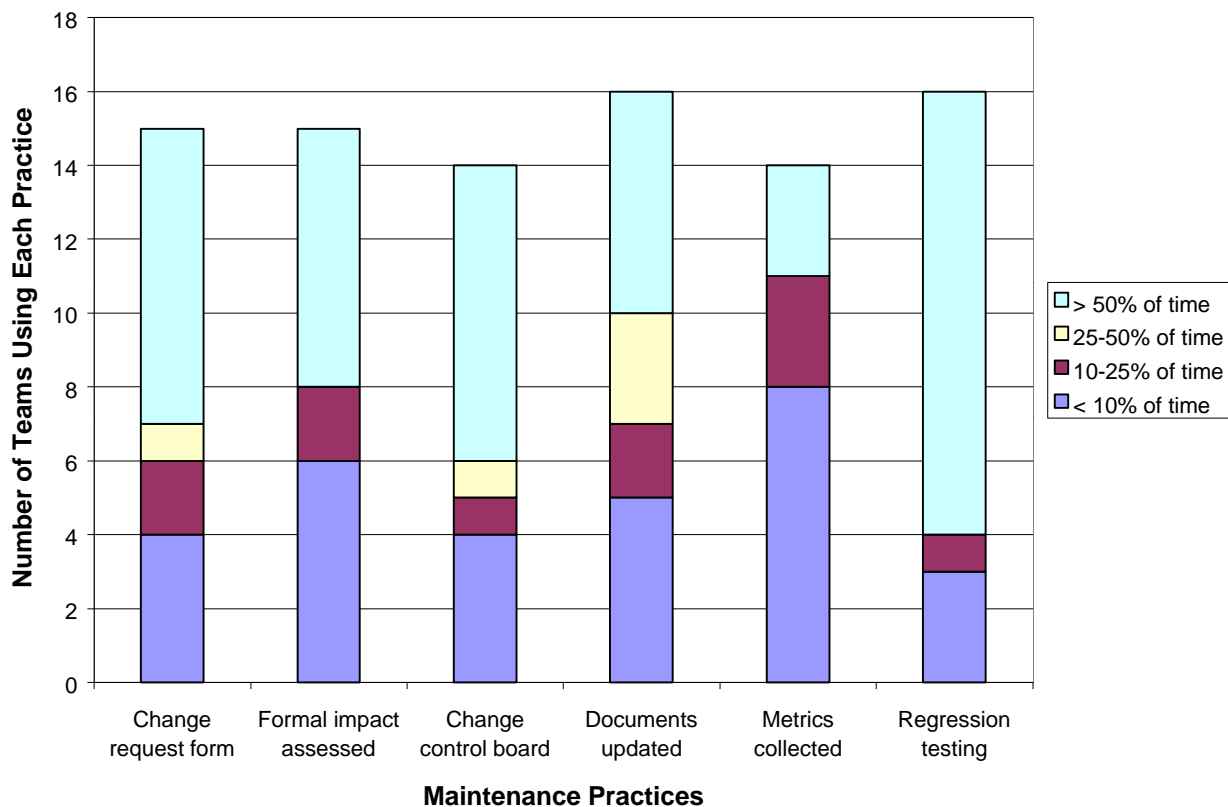


Figure 22. Use of Standard Maintenance Practices

It is apparent that, of these standard maintenance practices, only regression testing is consistently employed among the teams we surveyed. Change request forms and change control boards are used more than 50% of the time by more than 50% of the teams. Formal impact assessment is used more than 50% of the time by just under 50% of the teams. Updating of documents is divided, with roughly as many teams updating documents less than 10% of the time as update documents more than 50% of the time. Collection of metrics is firmly in last place, with most of the teams collecting metrics less than 10% of the time, if at all.

5.4 Other Issues

5.4.1 Software Training

Among the 20 teams surveyed, team members spent an average of 7.4 days per year in software-related training. The highest number of reported training days per team member was 30 (primarily self-taught on-the-job training {OJT}); the lowest was, 0. Because the teams that included contractor personnel operated in a “badgeless” manner, we did not differentiate between civil servant training and contractor training.

We also observed that, as noted in the prior profile reports (References 1, 2, and 6), the training offered to GSFC civil servant personnel is not generally viewed as an integrated set of software courses. In other words, the current software training is not considered a training program. Rather, software training courses are most often ad hoc and focus on specific software technologies. Nine teams reported that training was generally provided on an as-needed basis.

“As-needed” training, however, is not necessarily substandard. Two team leads indicated that they examined each team member’s current skill set and related it to current project needs to identify areas where training was needed. Four other team leads reported that they planned software training by looking at upcoming work, emerging

technologies, new products, or new systems that were expected to become available. Two teams reported that OJT comprised a significant part of the training provided.

Seven teams did indicate that they had a more structured approach to planning employee training. One team lead reported that she typically checked courses available on-site and used employees' Performance Plans to find courses of interest to them. Two team leads reported, "Training needs and plans are based on project development requirements." Two others noted that they planned training by comparing the skills of team members with the skills needed for the project. A sixth team lead reported, "We perform a post-mortem after each major software delivery to determine what skills are needed; ask each team lead to draft a list of training requirements and candidates; allocate monies from development budget; schedule the vendors (in-house and off-site)." Another said, "We developed a skills database to try and track what people know or do not know. Suggestions for specific training came from members of the group. Made use of existing [contractor] and GSFC courses."

5.4.2 Software Measures

Table 10 reports on collection of metrics. Out of 20 teams, seven routinely collect and analyze at least one type of software measure. Only one team collects and analyzes productivity metrics. Four teams collect and analyze defect measures.

Table 10. Collection of Metrics among the ISC Branches

	Collect and analyze routinely at least one type of metric	Collect routinely at least one type of metric	Do not collect any metrics
581			1
582	2	1	0
583	1		1
584	4		2
585			2
586			1
587			1
588			4

5.4.3 Software Engineering Research

We also explored the software engineering research projects currently underway within ISC. By software engineering research, we mean research projects that focus on advancing the state-of-the-art of software technologies. Seven of the 20 teams surveyed professed to be conducting some level of software engineering research. Three of these seven teams are in Code 584 and two are in Code 588. Codes 582 and 583 each have one team performing research.

The effort allocated to software engineering research among these seven teams varied from 1% to 40%. The average research effort among these seven teams was just under 10%; the average research effort among all 20 teams was 3.5%. Code 588 reported the highest level of software engineering research activity; the average research level reported among the four projects we studied in that code was 12%. Code 584 reported the second highest level of activity with an average of 4% allocated to research for the six projects studied. We did not ask the team leads for a description of the type of research they were conducting.

5.4.4 Process Improvement Activities

We found a significant level of interest in, and support for, process improvement activities throughout the ISC. Although there was no evidence of an integrated process improvement effort at the Center level, we found significant process improvement activities in many of the branches within ISC and in some of the specific teams that we surveyed. A wide variety of process improvement activities was reported, such as:

- Use of specific tools to increase efficiency,
- Documentation of current processes as a prelude to process improvement,
- Development and application of software metrics,
- Efforts to decrease cycle time,
- Establishment of an internet-based problem reporting system,
- Use of an automated system for tracking Discrepancy Reports,
- ISO 9000 registration,
- Configuration management,
- Increased reuse of software and simulators, and
- Increased use of COTS.

6 Recommendations

The objective of this study was to present an initial profile of the ISC, not to develop recommendations. Nevertheless, observations made during the study do suggest some specific conclusions and recommendations.

6.1 Recommendations

6.1.1 Training

The study team has some overall observations about training in general. We also have some specific recommendations regarding training in software languages.

As noted in section 5.4.1, we learned that the teams were spending an average of 7.4 days a year in training. This included all types of training: classroom training, individual study, and OJT. The 1994 GSFC profile (Reference 1) reported that civil servants across GSFC spent 5 days a year, on average, in software-related training, while contractor personnel averaged an additional day. The 1994 study doesn't specifically say whether or not these 5 or 6 days a year include OJT. Two of the respondents in the present ISC study, however, clearly stated that all or most of the training they cited was OJT. The 7.4-day average that includes OJT should therefore not be directly compared with the 1994 figures, which probably would have been higher if all of the respondents had included OJT. The 1994 Profile also found that the current software training was not considered a true training program (as defined in Section 1.5). The ISC may wish to consider developing a Center-wide software-training program; this would enable courses designed for a particular team or project to be made available to the entire Center.

Software development language and methodology are two specific areas where formal training may be desirable. In Section 4.3 we note that there is a clear trend away from well-established languages such as FORTRAN and C toward newer languages such as Java. We also note in Section 5.3.1 that *usage* of a methodology frequently exceeds *training* in that methodology. This suggests that training in these newer languages and development methodologies should be provided. We recommend that such training be made readily available to ISC software personnel in various formats: formal classes, textbooks, videotapes, and computer-based training, and that ISC personnel be encouraged to take advantage of this training.

6.1.2 COTS

In Section 4.4, we note that there is widespread use of COTS among the ISC teams. In view of this continuing trend, we recommend ongoing evaluation of COTS integration work. Careful, well-planned use of COTS can lead to significant cost savings; those teams that have delayed using COTS products may now want to consider doing so. It is vital for the ISC to remain aware of current software engineering research regarding COTS use and, in particular, cost estimation models for software including COTS products. We observed primarily horizontal reuse (traditional OS and language libraries, GUIs, DBMSs, and other general-purpose software). ISC might also consider heavier use of domain-oriented software products, such as ENVI (visualization of satellite data). Finally, the Center should initiate studies to define and evaluate models (e.g., development cost, maintenance cost, reliability curves), and recommended processes for COTS products.

6.1.3 Processes and Standards

In Section 5.1, we note that software processes and standards are generally established and defined at the project level. We believe that the ISC could realize significant process improvement if the respective branches were to examine the various processes and standards presently in use, evaluate them, and compare them against the "ISC Approved Team Processes for ISO 9001 Compliance" (Reference 8). ISC should then select the most promising, ISO-compliant processes and standards for establishment at the branch-level. After a suitable trial period (a year or two should be sufficient), ISC should then evaluate the branch-level software processes and standards, and select some for establishment (with possible tailoring) and trial at the Center level. This will enable the ISC to move towards an ISO-compliant set of processes, while still building on processes that are already in place.

6.1.4 Project Plans

In Section 5.1.3, we observe that only ten of the teams surveyed reported that they used project plans, and four of these ten teams reported that the project plan was followed but not maintained. We believe that a comprehensive and well-maintained project plan is a key success factor in both software development and maintenance. Accordingly, we strongly recommend that each branch adopt a standard format and template for a project plan, compliant with ISO 9001 and with the Product Plan Outline contained in the ISC Product Development Handbook (Reference 9). Each branch should then require that all projects within the branch prepare a project plan according to the adopted format, follow it, and maintain it. The regular updating of project documents in general would also be desirable, but the best place to begin, we believe, is with the project plan. We understand that software teams that produce products for missions or other operational applications are now addressing this. Teams that work in the domain of Advanced Technology Initiatives (e.g., Code 588) do not, as a rule, prepare project plans, perhaps because they are not subject to ISO 9001 registration. Code 588 may want to experiment with the use of project plans for a year, and then assess their effectiveness in the Advanced Technology environment.

6.1.5 Metrics

In Section 5.4.2, we report on the collection and analysis of metric data. W. Edwards Deming once observed that “You can’t improve what you can’t measure.” Collection and analysis of software measures are key to an effective process improvement program. Those same ISC teams that are developing project plans to meet ISO requirements have also been instructed by ISC management to begin collecting a minimum ISC-standard set of metrics. The SEL has begun working with the ISC in this effort by developing web-based data collection forms that can gather this data and store it in the SEL’s metric database, from where it can later be retrieved, analyzed, and reported.

Currently, ISC teams are collecting and storing their ISO metrics on a team level. We recommend that the teams test out the SEL data collection forms and consider using them to store their team metrics in the SEL database. We further recommend that each branch, in consultation with the SEL, expand on this minimal ISO-based measurement set, compare it with the NASA Core Metrics, and develop software measures appropriate to the branch’s needs. This basic set should include planned and actual milestone dates, project staffing by job category as a function of time, and system defects or problem reports. Eventually each branch should require each of its projects to begin collecting this set of branch-level metrics. The branch, and its constituent projects, should then begin analyzing the metric data to identify opportunities for process improvement.

Ultimately, some Center-wide standardization of software metrics would be desirable. The ISC should consider adopting the goal of developing a unified minimal set of software measures that will be collected across the entire Center. The Center should also establish a unified metrics database in which the measures for all ISC projects, teams, and domains can be collected. This will allow all ISC projects, teams, and domains to estimate and track cost and schedule in a consistent manner. The use of uniform metrics, and of a single unified metrics database, will also facilitate comparison of cost, schedule conformance, productivity, etc., across domains, teams, projects, and branches. This in turn will provide the basis for identifying and porting the most successful processes and techniques for use across a broader base. Each branch, project, team, and domain, of course, should be encouraged to augment the minimal set with additional metrics specific to its own environment. After a trial period (say, one year), the Center should conduct a review of all metrics to identify candidates for inclusion in the expanded minimal set. At the same time, the Center should also conduct a study of the collected measures to identify both successful processes and areas for process improvement.

6.2 Other Suggestions

In the ISC we have seen an importance placed on process and process improvement and an awareness of processes at a local level. There are several areas related to process improvement that the ISC ought to consider investing some effort.

6.2.1 Process Definition

One possible area for future work is process definition. Before an organization can improve its software engineering process, the process currently in use must be identified and described. Documentation of process is also a key prerequisite for ISO certification.

The present study was not intensive enough to generate the type of data that could be the basis for defining ISC processes. This is an initial baseline and, as such, we have addressed trends such as the use of COTS and new development languages. However, the study does suggest that there are specific parameters that may differentiate the processes used within the Center. These parameters are important factors to consider when either selecting or tailoring ISC processes. They are:

- *Size* - number of people on the team. Meaningful thresholds that could shape the behavior of the team might be 1, 10, 20.
- *Composition of the Team* – whether the team is composed primarily of contractors or civil servants. This influences the practices and the culture of the team.
- *Maintenance vs. Development* – whether the team is engaged mainly in new development or maintenance.
- *Operational vs. Advanced Technology Projects*. The type of project impacts the availability and stability of requirements, the need for plans, and the strictness of management practices.
- *Criticality of Project*. Mission critical projects, such as most Flight Software projects, impact development practices and choice of technologies.

On a lower level of detail, additional parameters to consider are development environment, development language, error causes, and use of COTS.

6.2.2 Process Improvement Initiatives

On a very small scale, individual teams could improve their processes and products by learning from both their own experiences and from similar project experiences. In the long run, a Center-wide improvement program is needed. The ISC can begin by using the processes already in place and supplementing them with other experiences where needed. This improvement program would likely begin both on a small scale with several teams, and for the entire ISC in some Center-wide initiatives.

6.2.3 ISO 9001 Registration

In order to maintain ISO 9001 registration, the ISC will need to address certain issues, such as metrics collection, process control, training, and basic process definition and documentation. As mentioned in Section 6.1, the SEL has begun working with the ISC in this effort, and plans are underway to collect data from software teams that produce a product for a mission or another type of operational usage.

6.3 Areas for Further Study

In Section 6.1.3, we suggested that the ISC study the standards and processes currently in use to identify candidates for standardization. We also proposed that the standards and processes currently in use within each ISC branch be studied to identify candidates for standardization at the branch level and, ultimately, at the Center level. The ISC should also consider similar studies in other areas where desirable activities could be adopted at the branch and Center levels. Two such areas are training and COTS integration. We propose that the training activities undertaken within teams, and the COTS integration methods (processes, tools, and models) that the teams use, be studied and evaluated for possible elevation to the branch or domain level. Then, after a suitable trial period, the training and COTS methods at the branch or domain level should be evaluated for possible adoption at the Center level.

Appendix A: ISC Baseline Branch-Level Interview Guide

This questionnaire is to be used to structure interviews with branch heads and associate branch heads in the ISC. The main objectives of these interviews are to characterize each branch in terms of high-level attributes and to identify all of the teams in each branch. The interviewer should feel free to modify the wording or order of the questions if it seems appropriate during the course of the interview. Also, questions should be skipped if they have already been addressed in enough detail earlier in the interview. The scribe's notes should follow this outline to facilitate data extraction.

Who: ISC Branch Heads and Associate Branch Heads

Subjects covered: Measurement Goal 1 of ISC Baseline Effort

Duration: 30-45 minutes

Interviewee:

ISC Branch:

Interviewer:

Scribe:

Date of interview:

Duration:

Location:

Introduction (general outline):

We're from the Software Engineering Laboratory, which is a group in Code 581 that studies software development projects in order to improve development practices in the local organization. The SEL also includes members from CSC and the University of Maryland [interviewer should indicate where they're from]. Up until recently, we've been working strictly with the Flight Dynamics Division (what used to be Code 550), but now our focus has shifted to the entire ISC. So one of our current projects is to better understand the software-related activities in the ISC by performing a baselining study.

More specifically, we would like to get a snapshot of the entire ISC organization at this point in time in terms of what kinds of work are being done, how the different branches are organized, what methods and techniques are being used, etc. This will give us a point of comparison against which to track future changes and improvements. It will also help the ISC management to understand the makeup of the Center and hopefully to identify areas where help is needed.

The purpose of this first interview is to get an overview of how your branch works. I'll be asking you some general questions about the type of work you do and how it's organized. Then I'll be giving you our questionnaire which asks for more detailed information, that you can fill out at your convenience over the next two weeks. This longer questionnaire is partially based on questionnaires used in past baselining efforts for Goddard and NASA as a whole. Then I'll be calling you to set up a time when we can sit down for a second interview where we'll be going over the questionnaire. Any questions before we start?

Question-1 *We've read your branch's functional description (dated June, 1997) on the Project Goddard Web page and it says [summarize in a sentence]. Is that still accurate?*

Question-2 *How is the work organized within the branch? In teams?*

Question-3 *Would you list for me all the teams in your branch?*

Question-4 *Is there any work going on in the branch that does not fall under a defined team [or other unit, depending on answer to question 2]? Are there defined leadership roles in teams (e.g. team leader, project leader, liaison, etc.)?*

Question-5 *How are the people in your branch funded (e.g. 100% full cost accounting)?*

Question-6 *How does outsourcing work in this branch? When is it used? For what types of work?*

Question-7 *What types of products does this branch produce?*

Question-8 *Who are customers and/or users of your products, in general?*

Question-9 *How do you see any of these characteristics of your branch changing in the short- or long-term future?*

Question-10 *What process improvement activities is your branch currently involved in?*

Appendix B: ISC Baseline Branch-Questionnaire

Name _____ Date _____

Branch Name _____ Position/Title _____

Telephone Number (____) _____ Email address _____

While accuracy is important, please don't spend a lot of time searching through personnel or other records. If you are guessing, just say so in the margin for that question. Your best estimates are probably adequate for our purposes. If you don't know an answer, just leave it blank. If an area doesn't apply to the types of work your Branch does, mark it "N/A" or write "None".

When you fill in this questionnaire, if you are not clear on the meaning of a term in a question, please leave it blank and ask your SEL representative during the follow-up interview. Unless the question says differently, your answers should encompass both your civil servants and your directly supporting contractors including subcontractors.

1. Overall Characteristics

a) What is the size of your Branch?
 Number of Full time equivalents (FTEs): _____ in-house _____ Contractors

b) What are the application domains of the work in your Branch?

c) What software projects are worked on in your branch?

(Please indicate if you have overall responsibility for that software project)

project name	is branch responsible?
_____	Yes _____ No _____
_____	Yes _____ No _____
_____	Yes _____ No _____
_____	Yes _____ No _____
_____	Yes _____ No _____
_____	Yes _____ No _____
_____	Yes _____ No _____
_____	Yes _____ No _____
_____	Yes _____ No _____

[In-house personnel assigned to software tasks are those involved 50% or more of their time in requirements, design, coding, testing, documenting, performing software quality assurance, conducting configuration control over the software, performing IV&V, and/or doing any aspect of software testing. Software personnel also encompasses up to and including the 2nd tier management responsible for the Team. "Software personnel" does not include staff providing computer center resources or other people with only peripheral involvement.]

d) How many in-house people in your Branch are working software tasks? _____ FTEs

e) How many contractor people in your Branch are working software tasks? _____ FTEs

Appendix B: ISC Baseline Branch-Questionnaire (cont'd)

[All questions beyond this point relate to software personnel or software work.]

- f) Do you have people in your branch matrixed to projects elsewhere **within ISC**? Yes____ No____
If yes, what percent are matrixed ____% FTEs
- g) Do you have people in your branch matrixed to projects **outside the ISC**? Yes____ No____
If yes, what percent are matrixed ____% FTEs
- h) What percent of the work in your branch is currently contracted? ____% of budget contracted
- i) List the companies which are the main contractors serving your branch:

- j) Do you have any work contracted out to another code? Yes____ No____
If yes, please specify GSFC CODE

[Note: Software development is defined to begin at the start of writing down software requirements. We define "software maintenance" as all activities that take place from the time of the first operational use of the first version of the software. That means software maintenance includes all fixing, enhancing, new versions, adapting, and changing that takes place after IOC regardless of how small or big the maintenance activity is.]

- k) What percent of effort is spent on the following:
Development ____% Maintenance ____% Other ____%.
(please specify _____)

For questions l-q, Answer for the major domains in your branch. If there are significant differences among domains discuss this with your SEL representative at the follow-up interview. The sum of all activities should total 100% for each domain.

- l)* What is the approximate allocation of effort of your Branch's software development (i.e., not maintenance) activities to the following activities? *(The sum of all activities should total 100%)*

Reqs/Analysis ____% Design ____% Coding ____% Testing ____% Other ____%
(please specify _____)

- m) Indicate the distribution of the branch personnel among the following roles:

(The sum of all activities should total 100%)

System Engineers ____% Programmers ____% Analysts ____% Testers ____%
Tech Leads ____% QA/CM ____% Managers ____% Other ____% (please specify _____)

Appendix B: ISC Baseline Branch-Questionnaire (cont'd)

- n)* What percentage of your software development budget is typically spent on:
(The sum of all activities should total 100%)
QA _____% Configuration Management _____% Documenting _____% Managing _____%
- o)* What was the approximate allocation of your Branch's maintenance activities to the following activities?
(The sum of all activities should total 100%)
Reqs/Analysis _____% Design _____% Coding _____% Testing _____% Other _____%
(please specify _____)
- p)* What percentage of your maintenance budget is typically spent on:
(The sum of all activities should total 100%)
QA _____% Configuration Management _____% Documenting _____% Managing _____%
- q)* What was the approximate allocation of that maintenance amount for:
(The sum of all activities should total 100%)
Correcting _____% Enhancing _____% Adapting _____% Testing/Verifying _____%
- r) Is your Branch currently doing any software engineering research? Yes____ No____
If yes:
1) what is the number of people currently involved in research
(in-house & contractor) that is budgeted? _____ FTEs
2) identify the technology being researched _____

2. How Much Software Do You Have?

- a) Approximately how many lines of code are operationally available (software you are responsible for maintaining) in your Branch in each of the following categories and what is the usual lifetime of the software in each category?

[By software "lifetime", we mean the time from first operational use of the first version of the software to the end of the last version's operational application.]

<u>Category</u>	<u>Lines of Code</u>	#Active Project	<u>Software Lifetime</u> [Use Check or %]			
			<2yrs	2-4yrs	4-7yrs	>7yrs
<i>Embedded</i>	_____	_____	_____	_____	_____	_____
<i>On-board Data Handling</i>	_____	_____	_____	_____	_____	_____
<i>Mission Ground Support</i>	_____	_____	_____	_____	_____	_____
<i>Information Mgmt Support</i>	_____	_____	_____	_____	_____	_____
<i>General Support</i>	_____	_____	_____	_____	_____	_____
<i>Off-line Data Systems</i>	_____	_____	_____	_____	_____	_____
<i>Science Processing</i>	_____	_____	_____	_____	_____	_____
<i>Administrative</i>	_____	_____	_____	_____	_____	_____
<i>Research</i>	_____	_____	_____	_____	_____	_____

Your use of "Lines of Code" (check one): Total physical lines _____
 Executable lines only _____
 Total non-comment lines _____
 Other lines _____

3. People Characteristics

- a) What is the average number of days per year software personnel in your Branch spend in software-related training?

_____ Days For in-house People _____ Days For Contractor People

["Software personnel" are defined as all personnel involved 50% or more of their time in requirements, design, coding, testing, documenting, performing software quality assurance, conducting configuration control over the software, performing IV&V, and/or doing any aspect of software testing. "Software personnel" also encompass up to and including the 2nd tier management responsible for the Team. "Software personnel" do not include staff providing computer center resources (e.g., tape mounters) or other people involved only peripherally with the software.]

- b) Does your Branch participate in a Software Training Program? Yes _____ No _____

[By Software Training Program, we mean a reasonably integrated set of related courses offered on a regular basis designed to maintain and improve the skills necessary to develop, manage, assure, and deliver quality software using modern, proven techniques.]

- c) Do your contractor organizations participate in a Software Training Program?

All do _____ Most do _____ Few do _____

- d) Is there a recommended software training program for the key software positions?

In-house Yes _____ No _____

Contractor Yes _____ No _____

[A recommended software training program would mean that combinations of training and experience would be key criteria in the staffing of your software Teams. We define "key software positions" as those such as Software Team Manager, Systems Analyst, Requirements Engineer, Integration & Test Manager, Software Configuration Manager, and Software Quality Assurance Manager.]

- e) What percent of your Branch's software staff have college degrees applicable to software development?

	% of Civil Servants	% of Contractors
Computer Science or Related	_____ % OBEs	_____ % OBEs
Other Technical [physics, engineering...]	_____ % OBEs	_____ % OBEs
Non-Technical Degree or No Degree	_____ % OBEs	_____ % OBEs

- f) What percent of your software team management falls into the following ranges of software management experience? [Note: S/W mgmt experience for any size s/w Team. Consider OBEs not FTEs]

In-house Team Managers	<3 yrs _____ %	3-5 yrs _____ %	5-10 yrs _____ %	> 10 yrs _____ %
Contractor Team Managers	<3 yrs _____ %	3-5 yrs _____ %	5-10 yrs _____ %	> 10 yrs _____ %

- g) How would you rate present software training activities from the standpoint of usefulness and applicability to your Branch's software work? (1 = not helpful; 5 = very helpful)

Available to in-house personnel	Rating _____
Available to contractor personnel	Rating _____

4. Software Processes Used for Developing and Maintaining Software

["Software process" means the phases, activities, and products by which the software is defined, developed, documented, and delivered. Such a process would include policies and standards, formal and informal reviews, and collection, analysis, and use of metrics.]

- a) What percentage of your Branch (including contractors) uses defined, written, advocated software processes? _____%
- b) To what extent are these software processes used? (Check one)
Minimal use ____ Some use ____ Extensive use ____
- c) How helpful are the software processes? (Check one)
Minimally helpful ____ Somewhat helpful ____ Very helpful ____
- d) To what degree are these software processes enforced? (Check one)
Minimally enforced ____ Somewhat enforced ____ Rigorously enforced ____
- e) Where are your software processes documented, and who owns them?

- f) What percentage of your Branch (including contractors) use software standards? _____%
- g) To what extent are these standards used? (Check one)
Minimal use ____ Some use ____ Extensive use ____
- h) How helpful are the standards? (Check one)
Minimally helpful ____ Somewhat helpful ____ Very helpful ____
- i) To what degree are these standards enforced? (Check one)
Minimally enforced ____ Somewhat enforced ____ Rigorously enforced ____
- j) What standards are used in your branch? NASA or other standards (e.g. ANSI, IEEE, ISO)

- k) Where are your software standards documented, and who owns them?

- l) Does your branch use Commercial Off-the-Shelf (COTS) products as components of deliverable systems (embedded COTS)? Yes____ No____
If yes, please specify

Appendix B: ISC Baseline Branch-Questionnaire (cont'd)

- m) Does your branch use COTS products to support software development and maintenance (that must be delivered with a system)? *Yes*____ *No*____

If yes, please specify

- n) Does your branch use COTS products to support software that is **NOT** delivered with system

*Yes*____ *No*____

If yes, please specify

- o) What languages is your Branch using for new software presently under development?

Fortran ____% Cobol ____% C ____% C++ ____% Ada ____% 4GL ____%

Other (specify): _____ %

_____ %

_____ %

- p) What percent of your Branch's **existing** software is written in the following languages?

Fortran ____% Cobol ____% C ____% C++ ____% Ada ____% 4GL ____%

Other (specify): _____ %

_____ %

_____ %

- q) What are the major testing techniques used in your Branch?

Are forms used to record test results? *Yes*____ *No*____

Is there training for testing? *Yes*____ *No*____

Is data archived? *Yes*____ *No*____

- r) What are the key documents produced and used by your process?

- s) What other non-software deliverables are produced and used by your process?

Appendix B: ISC Baseline Branch-Questionnaire (cont'd)

- t) Are Project Plans typically used by the projects run by your branch? *Yes*____ *No*____

[Note: "Project Plans" include Management and/or Development Plans]

If yes are the plans: (check the one that applies)

Kept Current & Followed____ Followed but NOT maintained____ NOT Followed NOR Maintained ____

- u) What types of tools are routinely applied by your Branch's software development Teams? (Use check marks)

Requirements Analysis _____ Traceability _____ Design/Graphics _____

Documentation _____ Debuggers _____ Test Data Generators _____

Test Coverage _____ QA Checkers _____ CM Aids _____

Complexity Measuring _____

Other Types (please list): _____

- v) What is the knowledge and usage in your Branch of the following:

	---- Awareness ----			---- Training ----			---- Usage ----		
	<i>Minimal</i>	<i>Some</i>	<i>Much</i>	<i>Minimal</i>	<i>Some</i>	<i>Much</i>	<i>Minimal</i>	<i>Some</i>	<i>Much</i>
<i>Prototyping</i>	_____	_____	_____	_____	_____	_____	_____	_____	_____
<i>Object-Oriented Technology</i>	_____	_____	_____	_____	_____	_____	_____	_____	_____
<i>Inspections/Walkthroughs</i>	_____	_____	_____	_____	_____	_____	_____	_____	_____
<i>Cleanroom Techniques</i>	_____	_____	_____	_____	_____	_____	_____	_____	_____
<i>Formal Methods</i>	_____	_____	_____	_____	_____	_____	_____	_____	_____
<i>CASE tools</i>	_____	_____	_____	_____	_____	_____	_____	_____	_____
<i>Structured Analysis</i>	_____	_____	_____	_____	_____	_____	_____	_____	_____
<i>Information Hiding</i>	_____	_____	_____	_____	_____	_____	_____	_____	_____
<i>COTS Integration</i>	_____	_____	_____	_____	_____	_____	_____	_____	_____
<i>Reliability Modeling</i>	_____	_____	_____	_____	_____	_____	_____	_____	_____
<i>Defect Causal Analysis</i>	_____	_____	_____	_____	_____	_____	_____	_____	_____

- w) Please respond concerning types of software metrics used in your Branch. [Software "measures" and software "metrics" are interchangeable terms.] Possible answers are: 1) Never, 2) Some, or 3) Routinely.

<u>Type</u>	<u>Data Collected?</u>	<u>Analyzed?</u>	<u>Feedback to Team?</u>	<u>Archived in a Database?</u>
Resource (effort, computer use...)	_____	_____	_____	_____
Defects (errors and their causes...)	_____	_____	_____	_____
Product (code size, pages of documentation...)	_____	_____	_____	_____
Process (extent of training, records of reviews...)	_____	_____	_____	_____
Productivity (Volume of work per unit of time, eg.,SLOC per staff years)	_____	_____	_____	_____
Project characteristics (language, platform)	_____	_____	_____	_____
Modifications (effort, reason, application domain, Team experience.)	_____	_____	_____	_____

- x) List the software measures that are collected by your branch:

_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

- y) At the start of preliminary design, are the software requirements typically stable and well understood?
 _____ Very stable _____ Fairly stable _____ Unstable

Appendix B: ISC Baseline Branch-Questionnaire (cont'd)

- z) Are requirements placed under rigorous, well-defined change control? *Yes*____ *No*____
If yes, when? _____ SRR? _____ PDR? _____ CDR? _____ Other (list)_____
- aa) Is change control discipline **maintained** according to known, **written** rules from SRR throughout the remainder of the software's life? *Yes*____ *No*____
- bb) After CDR, do software projects usually experience substantial (>25%) redesign? *Yes*____ *No*____

5. PRODUCT CHARACTERISTICS

- a) What types of software products are developed and maintained in your Branch?
(Check all that apply)

<i>Embedded</i>	_____
<i>On-board Data Handling</i>	_____
<i>Mission Ground Support</i>	_____
<i>Information Mgmt Support</i>	_____
<i>General Support</i>	_____
<i>Off-line Data Systems</i>	_____
<i>Science Processing</i>	_____
<i>Administrative</i>	_____
<i>Research</i>	_____
<i>Other (please specify)</i>	_____

- b) What is the defect rate in your operational software?

Number of Errors/kSLOC

Minimum Defect Rate	_____
Average Defect Rate	_____
Maximum Defect Rate	_____

[Note: kSLOCs are defined as the sum of all physical lines; i.e., executable, non-executable, and commentary.]

- c) What are typical causes of errors in your Branch's operational software?
(Please rank most to least significant, 1 = most significant)

<i>Misinterpreted Requirements</i>	_____
<i>Changing Requirements</i>	_____
<i>Missing Requirements</i>	_____
<i>Design Errors</i>	_____
<i>Interfaces</i>	_____
<i>Coding Errors</i>	_____
<i>Environment Problems</i>	_____

Thank you for taking the time to work on this questionnaire.

Now that you have gone through all the questions, please contact your SEL representative to confirm that you are ready for your follow-up interview.

If you are not sure who your SEL representative is, contact Amy Parra at 301.794.1298 or aparra@cscmail.csc.com.

Appendix C: ISC Baseline Team-Level Interview Guide

This questionnaire is to be used to structure interviews with team leads and key team members within each ISC branch. The main objectives of these interviews are to characterize each team in terms of high-level attributes: function, personnel, roles, etc. The interviewer should feel free to modify the wording or order of the questions if it seems appropriate during the course of the interview. Also, questions should be skipped if they have already been addressed in enough detail earlier in the interview. The scribe's notes should follow this outline to facilitate data extraction.

Who: ISC Team Leads and Key Team Members

Subjects covered: Measurement Goal 1 of ISC Baseline Effort

Duration: 30-45 minutes

Interviewee(s):

ISC Branch:

Team Name within Branch:

Interviewer:

Scribe:

Date of interview:

Duration:

Location:

Introduction (general outline):

We're from the Software Engineering Laboratory, which is a group within ISC that studies software development projects in order to improve development practices in the local organization. The SEL also includes members from CSC and the University of Maryland [interviewer should indicate where they're from]. Up until recently, we've been working strictly with the Flight Dynamics Division (what used to be Code 550), but now our focus has shifted to the entire ISC. So one of our current projects is to better understand the software-related activities in the ISC by performing a baselining study.

More specifically, we would like to get a snapshot of the entire ISC organization at this point in time in terms of what kinds of work are being done, how the different branches are organized, what methods and techniques are being used, etc. This will give us a point of comparison against which to track future changes and improvements. It will also help the ISC management to understand the makeup of the Center and hopefully to identify areas where help is needed.

We've already interviewed your branch head, _____ [and associate branch head(s), _____]. Now we're in the process of capturing information about the various teams within Code _____. The purpose of this first interview is to get an overview of what your team's function is, who belongs to the team, and what their roles of the various team members are. I'll be asking you some general questions about the type of work you do and how it's organized. At the end of the interview, I'll give you an opportunity to ask any questions you may have about the SEL and its role. Then I'll be giving you our questionnaire which asks for more detailed information, that you can fill out at your convenience over the next two weeks. This longer questionnaire is partially based on questionnaires used in past baselining efforts for Goddard and NASA as a whole. Then I'll be calling you to set up a time when we can sit down for a second interview where we'll be going over the questionnaire. I'll also check back with you to see how you're coming with the questionnaire, and whether I can clarify anything for you. Any questions before we start?

Question-1 *On what project or projects is your team working?*

Appendix C: ISC Baseline Team-Level Interview Guide (cont'd)

Question-2 *What function does your team perform for each project?*

Question-3 *How is the work of your team related to the mission of Code ____, and to the work of any other teams within your branch?*

Question-4 *How long has your team been in existence?*

Question-5 *How was this team assembled? Over what period of time?*

Question-6 *How is the team structured?*

Question-7 *What types of products or services does the team deliver?*

Question-8 *Who are the customers or end-users of your team's products or services?*

Question-9 *Through what mechanism(s) are the contractors (if any) funded, e.g., GSA, SLA? Which Code(s) fund the work?*

Appendix C: ISC Baseline Team-Level Interview Guide (cont'd)

Question-10 *How are deadlines and goals set for the team (by customers or branch management)?*

Question-11 *Every team has to deal with change. What types of changes (e.g., crises, changes in direction, new technologies, process changes, and configuration management problems) have you encountered, and how have you dealt with them?*

Question-12 *What types of changes do you anticipate that the team will encounter in the near future? This might involve, for example, team organization and staffing level, significant personnel changes, new business areas, anticipated business or technical challenges, new technologies to be assimilated, upcoming ISO 9001 registration or other types of changes.*

Question-13 *Is your team currently involved in any process improvement activities? If so, could you please tell us about them?*

Question-14 *Do you have any questions for us concerning the SEL and its role?*

Additional Notes

Appendix D: ISC Baseline Team-Level Questionnaire

Name _____ Date _____
Position/Title _____
Telephone Number () _____
Email address _____
Branch/Code _____
Team Name _____

Introduction and Directions:

If you don't understand a question, please leave it blank. Your SEL representative will clarify it during the interview.

While accuracy is important, please don't take a lot of time searching through personnel or other records. If you are guessing, just say so in the margin for that question. Your best estimates are probably adequate for our purposes. If you don't know an answer, just leave it blank. If a question doesn't apply to the work of your team, mark it "N/A" or write "None". Also, unless the question says otherwise, your answers should encompass both the civil servants and the contractor personnel on your team.

This questionnaire contains two parts. Part I contains high-level questions; please answer all the questions in Part I if you possibly can. Part II contains more specialized questions relating to software processes. Please answer those questions in Part II for which you have the information. Feel free to enter "Don't Know" for any questions in Part II that you cannot answer. If you can't answer any of the questions in Part II, simply put "Don't Know" at the top of Part II.

PART I. HIGH-LEVEL QUESTIONS

1. GENERAL INFORMATION

- a) How many people do you have on your team? _____ FTEs
- b) What organizations (e.g., your own branch, other branches, or contractor organizations) are represented on your team? How many of the people on the team come from each organization?

ORGANIZATION	NUMBER OF FTEs
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Appendix D: ISC Baseline Team-Level Questionnaire (cont'd)

- c) Could you please indicate the different roles that the members of the team perform?

TEAM ROLE	CHECK IF REPRESENTED ON TEAM
Team Lead	_____
Sub-Team Lead	_____
Scientist	_____
Systems Analyst	_____
Requirements Engineer	_____
Mathematician	_____
Programmer	_____
Webmaster	_____
Tester	_____
Quality Assurance	_____
Configuration Management	_____
Other (please list other roles below)	
_____	_____
_____	_____
_____	_____

- d) What was the previous GSFC code(s) for this team or project(s)? Code(s)

- e) Is this team part of a larger team? Yes _____ No _____

If so, please identify the larger team by its GSFC Code and team name, and indicate its size and function.

- f) Does this team support software development or maintenance? (If neither, skip to question k.)

Development _____ Maintenance _____ Both _____ Neither _____

- g) If you answered "Both" to question f, please indicate how the team's software effort is divided between development and maintenance. (The numbers should add up to 100 %).

Development _____ % Maintenance _____ %

- h) What is the approximate allocation of your team's software development (i.e., not maintenance) effort to the following activities? (*The sum of all activities should total 100%.*)

Reqs Analysis _____% Design _____% Coding _____% Testing _____%
Qual. Ass. _____% Conf. Mgt. _____% Documentation _____% Management _____%

- i) What is the approximate allocation of your team's maintenance effort to the following activities? (*The sum of all activities should total 100%.*)

Reqs Analysis _____% Design _____% Coding _____% Testing _____%
Qual. Ass. _____% Conf. Mgt. _____% Documentation _____% Management _____%

- j) What is the approximate allocation of that maintenance effort to the following components:

(*The sum of all activities should total 100%.*)

Correcting _____% Enhancing _____% Adapting _____% Testing/Verifying _____%

Appendix D: ISC Baseline Team-Level Questionnaire (cont'd)

k) What percent of the team's workload is allocated to software engineering research? _____ %

2. PERSONNEL CHARACTERISTICS

a) How many years of team or project leadership experience does the team lead have?

_____ years experience

b) What skills in management practices do the team members bring to the group? If one or more of the team members has a management skill that's not listed here, please feel free to add it.

MANAGEMENT SKILL	CHECK IF REPRESENTED ON TEAM
------------------	------------------------------

Project Planning	_____
Software Effort Estimation	_____
Risk Analysis	_____
Team Building	_____
Communication	_____
Organizing	_____
Tracking Work Performed	_____

Other (please list other management skills below)

_____	_____
_____	_____

c) What software technical skills do the team members bring to the group? If one or more of the team members has a software skill that's not listed here, please feel free to add it.

SOFTWARE SKILL	CHECK IF REPRESENTED ON TEAM
----------------	------------------------------

Requirements Analysis	_____
Software Size Estimation	_____
Software Design	_____
Project-Specific Languages	_____
Walkthroughs and Inspections	_____
Testing Methods	_____
Web Site Development	_____
S/W Configuration Management	_____
S/W Quality Assurance	_____

Other (please list other software technical skills below)

_____	_____
_____	_____
_____	_____

d) How many days are spent in software related training per work-year? Please enter the total number of days per work-year, on average, for each member of the team. _____ days/work-year/team member

Appendix D: ISC Baseline Team-Level Questionnaire (cont'd)

e) How do you plan training for the team? How do you determine what training each staff member will receive?

3. PRODUCTS DEVELOPED AND USED

a) Do you have any metrics other than kSLOC to measure software size? Yes ____ No ____

If so, please specify these metrics. _____

[Note: kSLOCs are defined as the sum of all physical lines; i.e., executable, non-executable, and commentary.]

b) What specific software products does the team **develop**? For each product, please supply the information called for in the table below.

PRODUCT NAME	FUNCTION	SIZE (kSLOC OR OTHER METRIC)	LANGUAGE	ANTICIPATED OPERATIONAL LIFE (YEARS)	EFFORT (STAFF-YEARS)

c) What specific software products does the team **maintain**? For each product, please supply the information called for in the table below.

PRODUCT NAME	FUNCTION	SIZE (kSLOC OR OTHER METRIC)	LANGUAGE	ANTICIPATED OPERATIONAL LIFE (YEARS)	EFFORT (STAFF-YEARS)

d) What percent of your typical deliverable software products consist of embedded COTS or GOTS?
_____ %

e) What percent of your typical deliverable software products consist of reused code? _____ %

[For this survey, "reuse" means reuse of source code only. Reused code is the sum of lines of code contained in modules which are used verbatim from another system or with very minor change; e.g., less than 10% change.]

4. HIGH-LEVEL PROCESS CHARACTERISTICS

["Software process" means the phases, activities, and products by which the software is defined, developed, documented, and delivered. Such a process would include policies and standards, formal and informal reviews, and collection, analysis, and use of metrics.]

- a) What percentage of your team (including any contractors) uses defined, written, advocated software processes?
_____ %
- b) To what extent are these software processes used? (Check one)
Minimal use ____ Some use ____ Extensive use ____
- c) How helpful are the software processes? (Check one)
Minimally helpful ____ Somewhat helpful ____ Very helpful ____
- d) To what degree are these software processes enforced? (Check one)
Minimally enforced ____ Somewhat enforced ____ Rigorously enforced ____
- e) Where are your software processes documented, and who owns them?

- f) What percentage of your team (including any contractors) use software standards? _____ %
- g) To what extent are these standards used? (Check one)
Minimal use ____ Some use ____ Extensive use ____
- h) How helpful are the standards? (Check one)
Minimally helpful ____ Somewhat helpful ____ Very helpful ____
- i) To what degree are these standards enforced? (Check one)
Minimally enforced ____ Somewhat enforced ____ Rigorously enforced ____
- j) What standards are used in your team? List NASA or other standards (e.g., ANSI, IEEE, and ISO).

- k) Where are your software standards documented, and who owns them?

- l) Does your team use Commercial Off-the-Shelf (COTS) products as components of deliverable systems (i.e., embedded COTS)? Yes ____ No ____

If yes, please specify:

Appendix D: ISC Baseline Team-Level Questionnaire (cont'd)

- m) Does your team use COTS products to support software development and maintenance (that must be delivered with a system)? Yes____ No____

If yes, please specify:

- n) Does your team use COTS products to support software development and maintenance that are **NOT** delivered with a system)? Yes____ No____

If yes, please specify:

- o) What languages is your team using for new software presently under development?

Fortran ____% Cobol ____% C ____% C++ ____% Ada ____% 4GL ____%

Other (specify): _____ %

- p) What percent of your team's **existing** software is written in the following languages?

Fortran ____% Cobol ____% C ____% C++ ____% Ada ____% 4GL ____%

Other (specify): _____ %

- q) Which of the following key project documents does your team generally produce?

PROJECT DOCUMENT

CHECK IF NORMALLY PRODUCED

Project Plan _____

Requirements Specification _____

Design Document _____

Test Plan _____

Quality Assurance Plan _____

Configuration Management Plan _____

User's Guide _____

Other (please list other project documents below)

- r) What are the major testing techniques used in your team?

Are forms used to record test results? Yes____ No____

Is there training for testing? Yes____ No____

Is data archived? Yes____ No____

Appendix D: ISC Baseline Team-Level Questionnaire (cont'd)

s) What types of tools are routinely applied by your team? (Use checkmarks)

<i>Requirements Analysis</i>	_____	<i>Traceability</i>	_____	<i>Design/Graphics</i>	_____
<i>Documentation</i>	_____	<i>Debuggers</i>	_____	<i>Test Data Generators</i>	_____
<i>Test Coverage</i>	_____	<i>QA Checkers</i>	_____	<i>CM Aids</i>	_____
<i>Complexity Measuring</i>	_____				

Other Types (please list): _____

t) What are the characteristics of your software development environment?

Development platform: Hardware _____ Operating System _____

Target platform: Hardware _____ Operating System _____

PART II. DETAILED INFORMATION ON SOFTWARE AND MANAGEMENT PROCESSES

[If you don't have any information for the questions in Part II, please check here: _____]

5. DETAILED SOFTWARE PROCESS CHARACTERISTICS

a) When changes are made to completed software units, are the following practices employed?

	<10%	10-25%	25-50%	>50% of the time
<i>Change Request Form</i>	_____	_____	_____	_____
<i>Formal Impact Assessed?</i>	_____	_____	_____	_____
<i>Change Control Board?</i>	_____	_____	_____	_____
<i>Documents Updated?</i>	_____	_____	_____	_____
<i>Metrics Collected?</i>	_____	_____	_____	_____
<i>Regression Testing?</i>	_____	_____	_____	_____

b) What is the knowledge and usage in your team of the following:

	---- Awareness ----			---- Training ----			---- Usage ----		
	Minimal	Some	Much	Minimal	Some	Much	Minimal	Some	Much
Prototyping	_____	_____	_____	_____	_____	_____	_____	_____	_____
Object-Oriented Methods	_____	_____	_____	_____	_____	_____	_____	_____	_____
Inspections/Walkthroughs	_____	_____	_____	_____	_____	_____	_____	_____	_____
Cleanroom Techniques	_____	_____	_____	_____	_____	_____	_____	_____	_____
Formal Methods	_____	_____	_____	_____	_____	_____	_____	_____	_____
CASE Tools	_____	_____	_____	_____	_____	_____	_____	_____	_____
Structured Analysis	_____	_____	_____	_____	_____	_____	_____	_____	_____
Information Hiding	_____	_____	_____	_____	_____	_____	_____	_____	_____
COTS Integration	_____	_____	_____	_____	_____	_____	_____	_____	_____
Other _____	_____	_____	_____	_____	_____	_____	_____	_____	_____

Appendix D: ISC Baseline Team-Level Questionnaire (cont'd)

c) Please indicate the types of software metrics used in your team. *[Software "measures" and software "metrics" are interchangeable terms.] Possible answers are: 1) Never, 2) Sometimes, or 3) Routinely.*

<u>Type</u>	<u>Data Collected ?</u>	<u>Analyzed ?</u>	<u>Feedback to Team?</u>	<u>Archived in a Database:</u>
Resource (effort, computer use...)	_____	_____	_____	_____
Defects (errors and their causes...)	_____	_____	_____	_____
Product (code size, pages of documentation...)	_____	_____	_____	_____
Process (extent of training, records of reviews...)	_____	_____	_____	_____
Productivity (Volume of work per unit of time, SLOC per staff years)	_____	_____	_____	_____
Project characteristics (language, platform)	_____	_____	_____	_____
Modifications (effort, reason, application domain, team experience.)	_____	_____	_____	_____

d) What is the typical productivity of your group (delivered kSLOCs per hour)?

e) List the software measures that are collected by your team, and indicate how each measure is captured (e.g., from inspection checklists, reported by inspection moderators, reported by each programmer.

f) List the estimates (e.g., cost, schedule, software size, performance) that are used within the team, and state the raw data from which each estimate is derived (e.g., requirements, SLOC, comparison with a previous system, prototype).

g) List any mathematical models (e.g., system performance, software size, cost, schedule, or work flow) that the team regularly employs, and indicate how each model is used.

Appendix D: ISC Baseline Team-Level Questionnaire (cont'd)

h) Which of the following software life cycle models does your team generally employ?

SOFTWARE LIFE CYCLE MODEL	CHECK IF NORMALLY EMPLOYED
---------------------------	----------------------------

Waterfall	_____
Modified Waterfall*	_____
Evolutionary Prototyping	_____
Incremental Development	_____
Evolutionary Development	_____
Package-Based Development	_____
Legacy System Maintenance	_____
Spiral	_____

Other (please list other software life cycle models below)

* Examples of modified waterfall SLCMs are waterfall with overlapping phases and waterfall with parallel subprojects.

i) Which (if any) of the following types of reviews does the team use?

	Used? (Yes/No)
Formal project reviews	_____
In-process reviews	_____
Walkthroughs	_____
Inspections	_____
Process audits	_____
Quality audits	_____
Management audits	_____

j) If you indicated in Question i that you use formal project reviews, please indicate which specific formal reviews are typically held.

	Used? (Yes/No)
System Requirements Review	_____
Preliminary Design Review	_____
Critical Design Review	_____
Functional Conf. Audit	_____
Physical Conf. Audit	_____
Operational Readiness Review	_____

Other (please list other formal reviews below)

6. PRODUCT CHARACTERISTICS

- a) What is the typical defect density (e.g. number of defects per kSLOC) in your delivered or operational software?

	<i>Number of Errors/kSLOC</i>
Minimum Defect Density	_____
Average Defect Density	_____
Maximum Defect Density	_____

[NOTE: kSLOCs are defined as the sum of all physical lines, i.e., executable, non-executable, and commentary. If you use another definition of kSLOCs (for example, executable lines only), or if you normally use a measure other than kSLOC, please provide this information here.]

- b) What are typical causes of errors in your team's operational software?
(Please rank most to least significant, 1 = most significant)

Misinterpreted Requirements	_____
Changing Requirements	_____
Missing Requirements	_____
Design Errors	_____
Interfaces	_____
Coding Errors	_____
Environment Problems	_____

- c) What is the most costly type of error to fix? _____

- d) Typically, how stable are the software requirements that your team receives?

Very stable _____ Fairly stable _____ Unstable _____

7. MANAGEMENT CHARACTERISTICS

- a) What types of project risks do you typically encounter?

	Encountered? (Yes/No)
Technical risks	_____
Schedule risks	_____
Cost risks	_____
Performance risks	_____
Quality risks	_____
Operability risks	_____

Other (please list other types of risks encountered below)

- b) What types of risk mitigation techniques do you employ?

Appendix D: ISC Baseline Team-Level Questionnaire (cont'd)

- c) Are Project Plans typically used by the project(s) your team supports? Yes _____ No _____
[Note: "Project Plans" include Management and/or Development Plans.]

If "Yes", are the plans: _____ *(check the one that applies)*
Kept Current & Followed _____ *Followed but NOT Maintained* _____ *NOT Followed NOR Maintained* _____

- e) What is the project schedule, by project activity? Please include both software and non-software tasks.

PROJECT ACTIVITY	PER CENT
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

- f) What is the approximate allocation of your project's total schedule to the following software activities? *(The sum of all activities should total 100%.)*
Reqs Analysis _____% Design _____% Coding _____% Testing _____% Other _____%

Appendix E: ISC Software System Development Team Questionnaire

[person's name here],

Your team has been selected by Marti Szczur to participate, in this survey. Please take a few minutes to fill out this simple questionnaire. Marti requested that you spend five minutes filling out this questionnaire and send us what you have at that time. Please make your best estimates (no need for intense calculations).

We appreciate that you are busy, and thank you for taking a few minutes to provide information about your team to the GSFC Software Engineering Laboratory (SEL). The SEL's mission is to (1) increase the understanding of the quality of software products & processes, (2) assess & refine software products & processes through studies, and (3) to use this knowledge to assist in implementation of processes and standards that are tailored to the environment.

Any questions contact: Dave Schultz, Software Engineering Laboratory (301) 805-3723

dschultz@csc.com

Team Name: _____

Team Lead: _____

Code: _____

Website URL for Team or Project Info: _____

SIZE

1. How many Full-Time Equivalents (FTE)s compose the team you lead? ____ Civil Servant ____ Contractors
2. How many people compose the team you lead (# of people)? ____ Civil Servant ____ Contractors
3. What percentage of Civil Servants from question 2 are matrixed from organizations outside ISC? ____%
4. What organizations do your matrixed people represent (from question3)?

_____	_____
_____	_____
_____	_____

CONTEXT OF WORK

5. What domains listed below categorize the type of work that your team does? (Please check one; if you support multiple domains select the primary domain)

____ Embedded Software Systems (such as flight s/w on board spacecraft)
____ Mission Ground Systems (such as flight dynamics, control center, command processing,)
____ Information Management Support (such as Integrated Financial Management System (IFMP))
____ Science Processing (such as science product generation, archive, retrieval)
____ Advanced Technology Initiatives (such as new techniques, prototypes, R&D)
____ other (please list) _____

LARGER TEAM

6. If your team is part of a larger team, please indicate the larger team name and lead.

_____	_____
-------	-------

SUB-TEAMS

7. If you manage individual sub-teams, and were not able to provide this information for those people, please indicate sub-team names and leads.

_____	_____
_____	_____
_____	_____
_____	_____

8. What percent of your work can be classified as Development ____%; Maintenance ____%

Additional Comments:

Abbreviations and Acronyms

4GL	Fourth-generation language
CASE	Computer-aided software engineering
C2P	Classification theory, consensus theory, and pattern recognition
CEO	Chief executive officer
CLIPS	C Language Integrated Production Systems
CM	Configuration management
COTS	Commercial off-the-shelf
CSC	Computer Sciences Corporation
DBMS	Database management system
FDD	Flight Dynamics Division
FTE	Full time equivalents
FY	Fiscal year
GREAS	Generic Resource, Event & Activity Scheduler
GSFC	Goddard Space Flight Center
GUI	Graphical user interface
HTML	Hypertext Media Language
IDL	Interface Development Language
IFMS	Integrated Financial Management System
ISC	Information Systems Center
ISO	International Organization for Standardization
IT	Information technology
IV&V	Independent verification and validation
MSLOC	Million source lines of code
NASA	National Aeronautics and Space Administration
NGST	Next Generation Space Telescope
OJT	On-the-job training
O-O	Object-oriented (methods)
OS	Operating system
R&D	Research and development
RT	Real-time
SEL	Software Engineering Laboratory
STK	Satellite Tool Kit
TCL	Tool Command Language
UM	University of Maryland
XML	Extensible Markup Language

References

1. D. Hall, F. McGarry, and C. Sinclair, *Profile of Software at the Goddard Space Flight Center*, NASA Software Engineering Program, NASA-RPT-002-94, June 1994, p. ES-1.
2. D. Hall, R. Pajerski, C. Sinclair, and B. Siegel, *Profile of Software at the National Aeronautics and Space Administration*, NASA Software Engineering Program, NASA-RPT-004-95, March 1995, p. ES-1.
3. *President's Information Technology Advisory Committee: Interim Report to the President*, August 1998, pp. 1, 21.
4. F. McGarry, G. Page, V. Basili, et al., *An Overview of the Software Engineering Laboratory*, NASA/GSFC Software Engineering Laboratory Series, SEL-94-005, December 1994.
5. Ibid., p. 5.
6. D. Hall and F. McGarry, *Profile of Software Within Code 500 at the Goddard Space Flight Center*, NASA Software Engineering Program, NASA-RPT-001-94, April 1994.
7. *Capability Maturity Model for Software (Version 1.1)*, Carnegie Mellon Software Engineering Institute, CMU/SEI-93-TR-024, 1993.
8. *ISC Approved Team Processes for ISO 9001 Compliance*, NASA/GSFC Information Systems Center, at <http://isc.gsfc.nasa.gov/Iso9k/atp/ATP.html>, August 1999.
9. *Information Systems Center Product Development Handbook*, NASA/GSFC Information Systems Center, at <http://isc.gsfc.nasa.gov/Iso9k/pdh/PDH.html>.

Standard Bibliography of SEL Literature

The technical papers, memorandums, and documents listed in this bibliography are organized into two groups. The first group is composed of documents issued by the Software Engineering Laboratory (SEL) during its research and development activities. The second group includes materials that were published elsewhere but pertain to SEL activities. The *Annotated Bibliography of Software Engineering Laboratory Literature* contains an abstract for each document and is available via the SEL Products Page at <http://sel.gsfc.nasa.gov/doc-st/docs/bibannot/contents.htm>.

SEL-ORIGINATED DOCUMENTS

- SEL-76-001, Proceedings From the First Summer Software Engineering Workshop, August 1976
- SEL-77-002, Proceedings From the Second Summer Software Engineering Workshop, September 1977
- SEL-78-005, Proceedings From the Third Summer Software Engineering Workshop, September 1978
- SEL-78-006, GSFC Software Engineering Research Requirements Analysis Study, P. A. Scheffer and C. E. Velez, November 1978
- SEL-78-007, Applicability of the Rayleigh Curve to the SEL Environment, T. E. Mapp, December 1978
- SEL-78-302, FORTRAN Static Source Code Analyzer Program (SAP) User's Guide (Revision 3), W. J. Decker, W. A. Taylor, et al., July 1986
- SEL-79-002, The Software Engineering Laboratory: Relationship Equations, K. Freburger and V. R. Basili, May 1979
- SEL-79-004, Evaluation of the Caine, Farber, and Gordon Program Design Language (PDL) in the Goddard Space Flight Center (GSFC) Code 580 Software Design Environment, C. E. Goorevich, A. L. Green, and W. J. Decker, September 1979
- SEL-79-005, Proceedings From the Fourth Summer Software Engineering Workshop, November 1979
- SEL-80-002, Multi-Level Expression Design Language-Requirement Level (MEDL-R) System Evaluation, W. J. Decker and C. E. Goorevich, May 1980
- SEL-80-005, A Study of the Musa Reliability Model, A. M. Miller, November 1980
- SEL-80-006, Proceedings From the Fifth Annual Software Engineering Workshop, November 1980
- SEL-80-007, An Appraisal of Selected Cost/Resource Estimation Models for Software Systems, J. F. Cook and F. E. McGarry, December 1980
- SEL-80-008, Tutorial on Models and Metrics for Software Management and Engineering, V. R. Basili, 1980
- SEL-81-011, Evaluating Software Development by Analysis of Change Data, D. M. Weiss, November 1981
- SEL-81-012, The Rayleigh Curve as a Model for Effort Distribution Over the Life of Medium Scale Software Systems, G. O. Picasso, December 1981
- SEL-81-013, Proceedings of the Sixth Annual Software Engineering Workshop, December 1981
- SEL-81-014, Automated Collection of Software Engineering Data in the Software Engineering Laboratory (SEL), A. L. Green, W. J. Decker, and F. E. McGarry, September 1981
- SEL-81-101, Guide to Data Collection, V. E. Church, D. N. Card, F. E. McGarry, et al., August 1982
- SEL-81-110, Evaluation of an Independent Verification and Validation (IV&V) Methodology for Flight Dynamics, G. Page, F. E. McGarry, and D. N. Card, June 1985
- SEL-81-305, Recommended Approach to Software Development, L. Landis, S. Waligora, F. E. McGarry, et al., June 1992
- SEL-81-305SP1, Ada Developers' Supplement to the Recommended Approach, R. Kester and L. Landis, November 1993

SEL-82-001, Evaluation of Management Measures of Software Development, G. Page, D. N. Card, and F. E. McGarry, September 1982, vols. 1 and 2

SEL-82-004, Collected Software Engineering Papers: Volume 1, July 1982

SEL-82-007, Proceedings of the Seventh Annual Software Engineering Workshop, December 1982

SEL-82-008, Evaluating Software Development by Analysis of Changes: The Data From the Software Engineering Laboratory, V. R. Basili and D. M. Weiss, December 1982

SEL-82-102, FORTRAN Static Source Code Analyzer Program (SAP) System Description (Revision 1), W. A. Taylor and W. J. Decker, April 1985

SEL-82-105, Glossary of Software Engineering Laboratory Terms, T. A. Babst, M. G. Rohleder, and F. E. McGarry, October 1983

SEL-82-1306, Annotated Bibliography of Software Engineering Laboratory Literature, D. Kistler, J. Bristow, and D. Smith, November 1994

SEL-83-001, An Approach to Software Cost Estimation, F. E. McGarry, G. Page, D. N. Card, et al., February 1984

SEL-83-002, Measures and Metrics for Software Development, D. N. Card, F. E. McGarry, G. Page, et al., March 1984

SEL-83-003, Collected Software Engineering Papers: Volume II, November 1983

SEL-83-007, Proceedings of the Eighth Annual Software Engineering Workshop, November 1983

SEL-83-106, Monitoring Software Development Through Dynamic Variables (Revision 1), C. W. Doerflinger, November 1989

SEL-84-003, Investigation of Specification Measures for the Software Engineering Laboratory (SEL), W. W. Agresti, V. E. Church, and F. E. McGarry, December 1984

SEL-84-004, Proceedings of the Ninth Annual Software Engineering Workshop, November 1984

SEL-84-101, Manager's Handbook for Software Development (Revision 1), L. Landis, F. E. McGarry, S. Waligora, et al., November 1990

SEL-85-001, A Comparison of Software Verification Techniques, D. N. Card, R. W. Selby, Jr., F. E. McGarry, et al., April 1985

SEL-85-002, Ada Training Evaluation and Recommendations From the Gamma Ray Observatory Ada Development Team, R. Murphy and M. Stark, October 1985

SEL-85-003, Collected Software Engineering Papers: Volume III, November 1985

SEL-85-004, Evaluations of Software Technologies: Testing, CLEANROOM, and Metrics, R. W. Selby, Jr. and V. R. Basili, May 1985

SEL-85-005, Software Verification and Testing, D. N. Card, E. Edwards, F. McGarry, and C. Antle, December 1985

SEL-85-006, Proceedings of the Tenth Annual Software Engineering Workshop, December 1985

SEL-86-001, Programmer's Handbook for Flight Dynamics Software Development, R. Wood and E. Edwards, March 1986

SEL-86-002, General Object-Oriented Software Development, E. Seidewitz and M. Stark, August 1986

SEL-86-003, Flight Dynamics System Software Development Environment (FDS/SDE) Tutorial, J. Buell and P. Myers, July 1986

SEL-86-004, Collected Software Engineering Papers: Volume IV, November 1986

SEL-86-005, Measuring Software Design, D. N. Card et al., November 1986

SEL-86-006, Proceedings of the Eleventh Annual Software Engineering Workshop, December 1986

SEL-87-001, Product Assurance Policies and Procedures for Flight Dynamics Software Development, S. Perry et al., March 1987

SEL-87-002, Ada[®] Style Guide (Version 1.1), E. Seidewitz et al., May 1987

SEL-87-003, Guidelines for Applying the Composite Specification Model (CSM), W. W. Agresti, June 1987

SEL-87-004, Assessing the Ada[®] Design Process and Its Implications: A Case Study, S. Godfrey, C. Brophy, et al., July 1987

SEL-87-009, Collected Software Engineering Papers: Volume V, November 1987

SEL-87-010, Proceedings of the Twelfth Annual Software Engineering Workshop, December 1987

SEL-88-001, System Testing of a Production Ada Project: The GRODY Study, J. Seigle, L. Esker, and Y. Shi, November 1988

SEL-88-002, Collected Software Engineering Papers: Volume VI, November 1988

SEL-88-003, Evolution of Ada Technology in the Flight Dynamics Area: Design Phase Analysis, K. Quimby and L. Esker, December 1988

SEL-88-004, Proceedings of the Thirteenth Annual Software Engineering Workshop, November 1988

SEL-88-005, Proceedings of the First NASA Ada User's Symposium, December 1988

SEL-89-002, Implementation of a Production Ada Project: The GRODY Study, S. Godfrey and C. Brophy, September 1989

SEL-89-004, Evolution of Ada Technology in the Flight Dynamics Area: Implementation/ Testing Phase Analysis, K. Quimby, L. Esker, L. Smith, M. Stark, and F. McGarry, November 1989

SEL-89-005, Lessons Learned in the Transition to Ada From FORTRAN at NASA/Goddard, C. Brophy, November 1989

SEL-89-006, Collected Software Engineering Papers: Volume VII, November 1989

SEL-89-007, Proceedings of the Fourteenth Annual Software Engineering Workshop, November 1989

SEL-89-008, Proceedings of the Second NASA Ada Users' Symposium, November 1989

SEL-89-103, Software Management Environment (SME) Concepts and Architecture (Revision 1), R. Hendrick, D. Kistler, and J. Valett, September 1992

SEL-89-301, Software Engineering Laboratory (SEL) Database Organization and User's Guide (Revision 3), L. Morusiewicz, February 1995

SEL-90-001, Database Access Manager for the Software Engineering Laboratory (DAMSEL) User's Guide, M. Buhler, K. Pumphrey, and D. Spiegel, March 1990

SEL-90-002, The Cleanroom Case Study in the Software Engineering Laboratory: Project Description and Early Analysis, S. Green et al., March 1990

SEL-90-003, A Study of the Portability of an Ada System in the Software Engineering Laboratory (SEL), L. O. Jun and S. R. Valett, June 1990

SEL-90-004, Gamma Ray Observatory Dynamics Simulator in Ada (GRODY) Experiment Summary, T. McDermott and M. Stark, September 1990

SEL-90-005, Collected Software Engineering Papers: Volume VIII, November 1990

SEL-90-006, Proceedings of the Fifteenth Annual Software Engineering Workshop, November 1990

SEL-91-001, Software Engineering Laboratory (SEL) Relationships, Models, and Management Rules, W. Decker, R. Hendrick, and J. Valett, February 1991

SEL-91-003, Software Engineering Laboratory (SEL) Ada Performance Study Report, E. W. Booth and M. E. Stark, July 1991

SEL-91-004, Software Engineering Laboratory (SEL) Cleanroom Process Model, S. Green, November 1991

SEL-91-005, Collected Software Engineering Papers: Volume IX, November 1991

SEL-91-006, Proceedings of the Sixteenth Annual Software Engineering Workshop, December 1991

SEL-91-102, Software Engineering Laboratory (SEL) Data and Information Policy (Revision 1), F. McGarry, August 1991

SEL-92-001, Software Management Environment (SME) Installation Guide, D. Kistler and K. Jeletic, January 1992

SEL-92-002, Data Collection Procedures for the Software Engineering Laboratory (SEL) Database, G. Heller, J. Valett, and M. Wild, March 1992

SEL-92-003, Collected Software Engineering Papers: Volume X, November 1992

SEL-92-004, Proceedings of the Seventeenth Annual Software Engineering Workshop, December 1992

SEL-93-001, Collected Software Engineering Papers: Volume XI, November 1993

SEL-93-002, Cost and Schedule Estimation Study Report, S. Condon, M. Regardie, M. Stark, et al., November 1993

SEL-93-003, Proceedings of the Eighteenth Annual Software Engineering Workshop, December 1993

SEL-94-001, Software Management Environment (SME) Components and Algorithms, R. Hendrick, D. Kistler, and J. Valett, February 1994

SEL-94-003, C Style Guide, J. Doland and J. Valett, August 1994

SEL-94-004, Collected Software Engineering Papers: Volume XII, November 1994

SEL-94-005, An Overview of the Software Engineering Laboratory, F. McGarry, G. Page, V. R. Basili, et al., December 1994

SEL-94-006, Proceedings of the Nineteenth Annual Software Engineering Workshop, December 1994

SEL-94-102, Software Measurement Guidebook (Revision 1), M. Bassman, F. McGarry, R. Pajerski, June 1995

SEL-95-001, Impact of Ada in the Flight Dynamics Division at Goddard Space Flight Center, S. Waligora, J. Bailey, M. Stark, March 1995

SEL-95-003, Collected Software Engineering Papers: Volume XIII, November 1995

SEL-95-004, Proceedings of the Twentieth Annual Software Engineering Workshop, December 1995

SEL-95-102, Software Process Improvement Guidebook (Revision 1), K. Jeletic, R. Pajerski, C. Brown, March 1996

SEL-96-001, Collected Software Engineering Papers: Volume XIV, October 1996

SEL-96-002, Proceedings of the Twenty-First Annual Software Engineering Workshop, December 1996

SEL-97-001, Guide To Software Engineering Laboratory Data Collection And Reporting, September 1997

SEL-97-002, Collected Software Engineering Papers: Volume XV, October 1997

SEL-97-003, Proceedings of the Twenty-Second Annual Software Engineering Workshop, December 1997

SEL-98-001, SEL COTS Study Phase 1 - Initial Characterization Study Report, A. Parra, August 1998

SEL-98-002, Proceedings of the Twenty-Third Annual Software Engineering Workshop, December 1998

SEL-RELATED LITERATURE

- ¹⁰Abd-El-Hafiz, S. K., V. R. Basili, and G. Caldiera, "Towards Automated Support for Extraction of Reusable Components", Proceedings of the IEEE Conference on Software Maintenance-1991 (CSM 91), October 1991
- ¹⁵Abd-El-Hafiz, S. K., V. R. Basili, "A Knowledge-Based Approach to the Analysis of Loops", IEEE Transactions on Software Engineering, May 1996
- ⁴Agresti, W. W., V. E. Church, D. N. Card, and P. L. Lo, "Designing With Ada for Satellite Simulation: A Case Study", Proceedings of the First International Symposium on Ada for the NASA Space Station, June 1986
- ²Agresti, W. W., F. E. McGarry, D. N. Card, et al., "Measuring Software Technology", Program Transformation and Programming Environments. New York: Springer-Verlag, 1984
- ¹Bailey, J. W. and V. R. Basili, "A Meta-Model for Software Development Resource Expenditures", Proceedings of the Fifth International Conference on Software Engineering. New York: IEEE Computer Society Press, 1981
- ⁸Bailey, J. W. and V. R. Basili, "Software Reclamation: Improving Post-Development Reusability", Proceedings of the Eighth Annual National Conference on Ada Technology, March 1990
- ¹⁰Bailey, J. W. and V. R. Basili, "The Software-Cycle Model for Re-Engineering and Reuse", Proceedings of the ACM Tri-Ada 91 Conference, October 1991
- ¹Basili, V. R., "Models and Metrics for Software Management and Engineering", ASME Advances in Computer Technology, January 1980, vol. 1
- Basili, V. R., Tutorial on Models and Metrics for Software Management and Engineering. New York: IEEE Computer Society Press, 1980 (also designated SEL-80-008)
- ³Basili, V. R., "Quantitative Evaluation of Software Methodology", Proceedings of the First Pan-Pacific Computer Conference, September 1985
- ⁷Basili, V. R., Maintenance = Reuse-Oriented Software Development, University of Maryland, Technical Report TR-2244, May 1989
- ⁷Basili, V. R., Software Development: A Paradigm for the Future, University of Maryland, Technical Report TR-2263, June 1989
- ⁸Basili, V. R., "Viewing Maintenance of Reuse-Oriented Software Development", IEEE Software, January 1990
- ¹³Basili, V. R., "The Experience Factory and Its Relationship to Other Quality Approaches", Advances in Computers, vol. 41, Academic Press, Incorporated, 1995
- ¹⁴Basili, V. R., "Evolving and Packaging Reading Technologies", Proceedings of the Third International Conference on Achieving Quality in Software, Florence, Italy, January 1996
- ¹⁴Basili, V. R., "The Role of Experimentation in Software Engineering: Past, Current, and Future", Proceedings of the Eighteenth Annual Conference on Software Engineering (ICSE-18), March 1996
- ¹⁶Basili, V. R., "Evolving and Packaging Reading Technologies", Journal of Systems and Software, 1997, 38: 3-12
- ¹Basili, V. R. and J. Beane, "Can the Parr Curve Help With Manpower Distribution and Resource Estimation Problems?", Journal of Systems and Software, February 1981, vol. 2, no. 1
- ¹³Basili, V. R., L. Briand, and W. L. Melo, A Validation of Object-Oriented Design Metrics, University of Maryland, Computer Science Technical Report, CS-TR-3443, UMIACS-TR-95-40, April 1995
- ¹⁵Basili, V. R., L. C. Briand and W. L. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators", IEEE Transactions on Software Engineering, October 1996

- ¹⁵Basili, V. R., L. C. Briand, and W. L. Melo, "How Reuse Influences Productivity in Object-Oriented Systems, Communications of the ACM, October 1996
- ¹⁴Basili, V. R., G. Calavaro, G. Iazeolla, "Simulation Modeling of Software Development Processes", 7th European Simulation Symposium (ESS '95), October 1995
- ¹³Basili, V. R. and G. Caldiera, The Experience Factory Strategy and Practice, University of Maryland, Computer Science Technical Report, CS-TR-3483, UMIACS-TR-95-67, May 1995
- ⁹Basili, V. R., G. Caldiera, and G. Cantone, "A Reference Architecture for the Component Factory", ACM Transactions on Software Engineering and Methodology, January 1992
- ¹⁰Basili, V. R., G. Caldiera, F. McGarry, et al., "The Software Engineering Laboratory—An Operational Software Experience Factory", Proceedings of the Fourteenth International Conference on Software Engineering (ICSE 92), May 1992
- ¹⁵Basili, V. R., S. E. Condon, K. El Emam, R. B. Hendrick and W. L. Melo, "Characterizing and Modeling the Cost of Rework in a Library of Reusable Software Components", International Conference on Software Engineering (ICSE-19), May 1997
- ¹Basili, V. R. and K. Freburger, "Programming Measurement and Estimation in the Software Engineering Laboratory", Journal of Systems and Software, February 1981, vol. 2, no. 1
- ¹²Basili, V. R. and S. Green, "Software Process Evolution at the SEL", IEEE Software, July 1994, pp. 58 - 66
- ¹⁴Basili, V. R., S. Green, O. Laitenberger, F. Shull, S. Sorumgard, and M. Zelkowitz, "The Empirical Investigation of Perspective-Based Reading", University of Maryland, Computer Science Technical Report, CS-TR-3585, UMIACS-TR-95-127, December 1995
- ³Basili, V. R. and N. M. Panlilio-Yap, "Finding Relationships Between Effort and Other Variables in the SEL", Proceedings of the International Computer Software and Applications Conference, October 1985
- ⁴Basili, V. R. and D. Patnaik, A Study on Fault Prediction and Reliability Assessment in the SEL Environment, University of Maryland, Technical Report TR-1699, August 1986
- ²Basili, V. R. and B. T. Perricone, "Software Errors and Complexity: An Empirical Investigation", Communications of the ACM, January 1984, vol. 27, no. 1
- ¹Basili, V. R. and T. Phillips, "Evaluating and Comparing Software Metrics in the Software Engineering Laboratory", Proceedings of the ACM SIGMETRICS Symposium/Workshop: Quality Metrics, March 1981
- ³Basili, V. R. and C. L. Ramsey, "ARROWSMITH-P—A Prototype Expert System for Software Engineering Management", Proceedings of the IEEE/MITRE Expert Systems in Government Symposium, October 1985
- Basili, V. R. and J. Ramsey, Structural Coverage of Functional Testing, University of Maryland, Technical Report TR-1442, September 1984
- Basili, V. R. and R. Reiter, "Evaluating Automatable Measures for Software Development", Proceedings of the Workshop on Quantitative Software Models for Reliability, Complexity, and Cost. New York: IEEE Computer Society Press, 1979
- ⁵Basili, V. R. and H. D. Rombach, "Tailoring the Software Process to Project Goals and Environments", Proceedings of the 9th International Conference on Software Engineering, March 1987
- ⁵Basili, V. R. and H. D. Rombach, "TAME: Tailoring an Ada Measurement Environment", Proceedings of the Joint Ada Conference, March 1987
- ⁵Basili, V. R. and H. D. Rombach, "TAME: Integrating Measurement Into Software Environments", University of Maryland, Technical Report TR-1764, June 1987

- ⁶Basili, V. R. and H. D. Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments", IEEE Transactions on Software Engineering, June 1988
- ⁷Basili, V. R. and H. D. Rombach, Towards A Comprehensive Framework for Reuse: A Reuse-Enabling Software Evolution Environment, University of Maryland, Technical Report TR-2158, December 1988
- ⁸Basili, V. R. and H. D. Rombach, Towards A Comprehensive Framework for Reuse: Model-Based Reuse Characterization Schemes, University of Maryland, Technical Report TR-2446, April 1990
- ⁹Basili, V. R. and H. D. Rombach, "Support for Comprehensive Reuse", Software Engineering Journal, September 1991
- ³Basili, V. R. and R. W. Selby, Jr., "Calculation and Use of an Environment's Characteristic Software Metric Set", Proceedings of the Eighth International Conference on Software Engineering. New York: IEEE Computer Society Press, 1985
- Basili, V. R. and R. W. Selby, "Comparing the Effectiveness of Software Testing Strategies", IEEE Transactions on Software Engineering, December 1987
- ³Basili, V. R. and R. W. Selby, Jr., "Four Applications of a Software Data Collection and Analysis Methodology", Proceedings of the NATO Advanced Study Institute, August 1985
- ⁵Basili, V. R. and R. Selby, "Comparing the Effectiveness of Software Testing Strategies", IEEE Transactions on Software Engineering, December 1987
- ⁹Basili, V. R. and R. W. Selby, "Paradigms for Experimentation and Empirical Studies in Software Engineering", Reliability Engineering and System Safety, January 1991
- ⁴Basili, V. R., R. W. Selby, Jr., and D. H. Hutchens, "Experimentation in Software Engineering", IEEE Transactions on Software Engineering, July 1986
- ²Basili, V. R., R. W. Selby, and T. Phillips, "Metric Analysis and Data Validation Across FORTRAN Projects", IEEE Transactions on Software Engineering, November 1983
- ²Basili, V. R. and D. M. Weiss, A Methodology for Collecting Valid Software Engineering Data, University of Maryland, Technical Report TR-1235, December 1982
- ³Basili, V. R. and D. M. Weiss, "A Methodology for Collecting Valid Software Engineering Data", IEEE Transactions on Software Engineering, November 1984
- ¹Basili, V. R. and M. V. Zelkowitz, "The Software Engineering Laboratory: Objectives", Proceedings of the Fifteenth Annual Conference on Computer Personnel Research, August 1977
- Basili, V. R. and M. V. Zelkowitz, "Designing a Software Measurement Experiment", Proceedings of the Software Life Cycle Management Workshop, September 1977
- ¹Basili, V. R. and M. V. Zelkowitz, "Operation of the Software Engineering Laboratory", Proceedings of the Second Software Life Cycle Management Workshop, August 1978
- ¹Basili, V. R. and M. V. Zelkowitz, "Measuring Software Development Characteristics in the Local Environment", Computers and Structures, August 1978, vol. 10
- Basili, V. R. and M. V. Zelkowitz, "Analyzing Medium Scale Software Development", Proceedings of the Third International Conference on Software Engineering, New York: IEEE Computer Society Press, 1978
- ¹³Basili, V. R., M. Zelkowitz, F. McGarry, G. Page, S. Waligora, and R. Pajerski, "SEL's Software Process-Improvement Program", IEEE Software, vol. 12, no. 6, November 1995, pp. 83 - 87
- ¹²Bassman, M. J., F. McGarry, and R. Pajerski, Software Measurement Guidebook, NASA-GB-001-94, Software Engineering Program, July 1994

- ⁹Booth, E. W. and M. E. Stark, "Designing Configurable Software: COMPASS Implementation Concepts", Proceedings of Tri-Ada 1991, October 1991
- ¹⁰Booth, E. W. and M. E. Stark, "Software Engineering Laboratory Ada Performance Study—Results and Implications", Proceedings of the Fourth Annual NASA Ada User's Symposium, April 1992
- ¹⁰Briand, L. C. and V. R. Basili, "A Classification Procedure for the Effective Management of Changes During the Maintenance Process", Proceedings of the 1992 IEEE Conference on Software Maintenance (CSM 92), November 1992
- ¹⁰Briand, L. C., V. R. Basili, and C. J. Hetmanski, "Providing an Empirical Basis for Optimizing the Verification and Testing Phases of Software Development", Proceedings of the Third IEEE International Symposium on Software Reliability Engineering (ISSRE 92), October 1992
- ¹¹Briand, L. C., V. R. Basili, and C. J. Hetmanski, Developing Interpretable Models with Optimized Set Reduction for Identifying High Risk Software Components, University of Maryland, Technical Report TR-3048, March 1993
- ¹²Briand, L. C., V. R. Basili, Y. Kim, and D. R. Squier, "A Change Analysis Process to Characterize Software Maintenance Projects", Proceedings of the International Conference on Software Maintenance, Victoria, British Columbia, Canada, September 19 - 23, 1994, pp. 38 - 49
- ⁹Briand, L. C., V. R. Basili, and W. M. Thomas, A Pattern Recognition Approach for Software Engineering Data Analysis, University of Maryland, Technical Report TR-2672, May 1991
- ¹⁴Briand, L., V. R. Basili, S. Condon, Y. Kim, W. Melo and J. D. Valett, "Understanding and Predicting the Process of Software Maintenance Releases", Proceedings of the Eighteenth Annual Conference on Software Engineering (ICSE-18), March 1996
- ¹⁴Briand, L., Y. Kim, W. Melo, C. B. Seaman, V. R. Basili, "Qualitative Analysis for Maintenance Process Assessment", University of Maryland, Computer Science Technical Report, CS-TR-3592, UMIACS-TR-96-7, January 1996
- ¹⁶Briand, L., Y. Kim, W. Melo, C. Seaman, and V. R. Basili, Q-MOPP: Qualitative Evaluation of Maintenance Organizations, Processes and Products, Journal of Software Maintenance: Research and Practice, 10, 249 - 278 1998
- ¹³Briand, L., W. Melo, C. Seaman, and V. R. Basili, "Characterizing and Assessing a Large-Scale Software Maintenance Organization", Proceedings of the 17th International Conference on Software Engineering, Seattle, Washington, U.S.A., April 23 - 30, 1995
- ¹¹Briand, L. C., S. Morasca, and V. R. Basili, "Measuring and Assessing Maintainability at the End of High Level Design", Proceedings of the 1993 IEEE Conference on Software Maintenance (CSM 93), November 1993
- ¹²Briand, L., S. Morasca, and V. R. Basili, Defining and Validating High-Level Design Metrics, University of Maryland, Computer Science Technical Report, CS-TR-3301, UMIACS-TR-94-75, June 1994
- ¹³Briand, L., S. Morasca, and V. R. Basili, Property-Based Software Engineering Measurement, University of Maryland, Computer Science Technical Report, CS-TR-3368, UMIACS-TR-94-119, November 1994
- ¹³Briand, L., S. Morasca, and V. R. Basili, Goal-Driven Definition of Product Metrics Based on Properties, University of Maryland, Computer Science Technical Report, CS-TR-3346, UMIACS-TR-94-106, December 1994
- ¹⁵Briand, L., S. Morasca, and V. R. Basili, "Property-Based Software Engineering Measurement, IEEE Transactions on Software Engineering, January 1996
- ¹⁵Briand, L., S. Morasca, and V. R. Basili, Response to: Comments on "Property-Based Software Engineering Measurement: Refining the Additivity Properties", IEEE Transactions on Software Engineering, March 1997
- ¹¹Briand, L. C., W. M. Thomas, and C. J. Hetmanski, "Modeling and Managing Risk Early in Software Development", Proceedings of the Fifteenth International Conference on Software Engineering (ICSE 93), May 1993

- ⁵Brophy, C. E., W. W. Agresti, and V. R. Basili, "Lessons Learned in Use of Ada-Oriented Design Methods", Proceedings of the Joint Ada Conference, March 1987
- ⁶Brophy, C. E., S. Godfrey, W. W. Agresti, and V. R. Basili, "Lessons Learned in the Implementation Phase of a Large Ada Project", Proceedings of the Washington Ada Technical Conference, March 1988
- ²Card, D. N., "Early Estimation of Resource Expenditures and Program Size", Computer Sciences Corporation, Technical Memorandum, June 1982
- ²Card, D. N., "Comparison of Regression Modeling Techniques for Resource Estimation", Computer Sciences Corporation, Technical Memorandum, November 1982
- ³Card, D. N., "A Software Technology Evaluation Program", Annais do XVIII Congresso Nacional de Informatica, October 1985
- ⁵Card, D. N. and W. W. Agresti, "Resolving the Software Science Anomaly", Journal of Systems and Software, 1987
- ⁶Card, D. N. and W. W. Agresti, "Measuring Software Design Complexity", Journal of Systems and Software, June 1988
- ⁴Card, D. N., V. E. Church, and W. W. Agresti, "An Empirical Study of Software Design Practices", IEEE Transactions on Software Engineering, February 1986
- Card, D. N., V. E. Church, W. W. Agresti, and Q. L. Jordan, "A Software Engineering View of Flight Dynamics Analysis System", Parts I and II, Computer Sciences Corporation, Technical Memorandum, February 1984
- Card, D. N., Q. L. Jordan, and V. E. Church, "Characteristics of FORTRAN Modules", Computer Sciences Corporation, Technical Memorandum, June 1984
- ⁵Card, D. N., F. E. McGarry, and G. T. Page, "Evaluating Software Engineering Technologies", IEEE Transactions on Software Engineering, July 1987
- ³Card, D. N., G. T. Page, and F. E. McGarry, "Criteria for Software Modularization", Proceedings of the Eighth International Conference on Software Engineering. New York: IEEE Computer Society Press, 1985
- ¹Chen, E. and M. V. Zelkowitz, "Use of Cluster Analysis To Evaluate Software Engineering Methodologies", Proceedings of the Fifth International Conference on Software Engineering. New York: IEEE Computer Society Press, 1981
- ⁴Church, V. E., D. N. Card, W. W. Agresti, and Q. L. Jordan, "An Approach for Assessing Software Prototypes", ACM Software Engineering Notes, July 1986
- ¹⁵Condon, S., R. Hendrick, M. E. Stark, W. Steger, "The Generalized Support Software (GSS) Domain Engineering Process: An Object-Oriented Implementation and Reuse Success at Goddard Space Flight Center", Addendum to the Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA 96), San Jose, California, U.S.A., October 1996
- ¹⁵Devanbu, P., S. Karstu, W. L. Melo and W. Thomas, "Analytical and Empirical Evaluation of Software Reuse Metrics, Proceedings of the 18th International Conference on Software Engineering (ICSE-18), March 1996
- ²Doerflinger, C. W. and V. R. Basili, "Monitoring Software Development Through Dynamic Variables", Proceedings of the Seventh International Computer Software and Applications Conference. New York: IEEE Computer Society Press, 1983
- Doubleday, D., ASAP: An Ada Static Source Code Analyzer Program, University of Maryland, Technical Report TR-1895, August 1987 (NOTE: 100 pages long)
- ⁶Godfrey, S. and C. Brophy, "Experiences in the Implementation of a Large Ada Project", Proceedings of the 1988 Washington Ada Symposium, June 1988

- ⁵Jeffery, D. R. and V. R. Basili, Characterizing Resource Data: A Model for Logical Association of Software Data, University of Maryland, Technical Report TR-1848, May 1987
- ⁶Jeffery, D. R. and V. R. Basili, "Validating the TAME Resource Data Model", Proceedings of the Tenth International Conference on Software Engineering, April 1988
- ¹⁶Kontio, J., The Riskit Method for Software Risk Management, Version 1.00, University of Maryland, Computer Science Technical Report, CS-TR-3782, UMIACS-TR-97-38, (Date)
- ¹⁶Kontio, J. and V. R. Basili, Empirical Evaluation of a Risk Management Method, SEI Conference on Risk Management, 1997
- ¹⁵Kontio, J., G. Caldiera, and V. R. Basili, "Defining Factors, Goals and Criteria for Reusable Component Evaluation", CASCON '96 Conference, November 1996
- ¹⁵Kontio, J., H. Englund, and V. R. Basili, Experiences from an Exploratory Case Study with a Software Risk Management Method, Computer Science Technical Report, CS-TR-3705, UMIACS-TR-96-75, August 1996
- ¹⁵Lanubile, F., "Why Software Reliability Predictions Fail", IEEE Software, July 1996
- ¹⁶Lanubile, F., "Empirical Evaluation of Software Maintenance Technologies", Empirical Software Engineering, 2, 97-108, 1997
- ¹¹Li, N. R. and M. V. Zelkowitz, "An Information Model for Use in Software Management Estimation and Prediction", Proceedings of the Second International Conference on Information Knowledge Management, November 1993
- ⁵Mark, L. and H. D. Rombach, A Meta Information Base for Software Engineering, University of Maryland, Technical Report TR-1765, July 1987
- ⁶Mark, L. and H. D. Rombach, "Generating Customized Software Engineering Information Bases From Software Process and Product Specifications", Proceedings of the 22nd Annual Hawaii International Conference on System Sciences, January 1989
- ⁵McGarry, F. E. and W. W. Agresti, "Measuring Ada for Software Development in the Software Engineering Laboratory (SEL)", Proceedings of the 21st Annual Hawaii International Conference on System Sciences, January 1988
- ⁷McGarry, F., L. Esker, and K. Quimby, "Evolution of Ada Technology in a Production Software Environment", Proceedings of the Sixth Washington Ada Symposium (WADAS), June 1989
- ¹³McGarry, F., R. Pajerski, G. Page, et al., Software Process Improvement in the NASA Software Engineering Laboratory, Carnegie-Mellon University, Software Engineering Institute, Technical Report CMU/SEI-94-TR-22, ESC-TR-94-022, December 1994
- ³McGarry, F. E., J. Valett, and D. Hall, "Measuring the Impact of Computer Resource Quality on the Software Development Process and Product", Proceedings of the Hawaiian International Conference on System Sciences, January 1985
- ¹⁵Morasca, S., L. C. Briand, V. R. Basili, E. J. Weyuker and M. V. Zelkowitz, Comments on "Towards a Framework for Software Measurement Validation", IEEE Transactions on Software Engineering, March 1997
- ³Page, G., F. E. McGarry, and D. N. Card, "A Practical Experience With Independent Verification and Validation", Proceedings of the Eighth International Computer Software and Applications Conference, November 1984
- ¹²Porter, A. A., L. G. Votta, Jr., and V. R. Basili, Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment, University of Maryland, Technical Report TR-3327, July 1994
- ⁵Ramsey, C. L. and V. R. Basili, "An Evaluation of Expert Systems for Software Engineering Management", IEEE Transactions on Software Engineering, June 1989

- ³Ramsey, J. and V. R. Basili, "Analyzing the Test Process Using Structural Coverage", Proceedings of the Eighth International Conference on Software Engineering. New York: IEEE Computer Society Press, 1985
- ⁵Rombach, H. D., "A Controlled Experiment on the Impact of Software Structure on Maintainability", IEEE Transactions on Software Engineering, March 1987
- ⁸Rombach, H. D., "Design Measurement: Some Lessons Learned", IEEE Software, March 1990
- ⁹Rombach, H. D., "Software Reuse: A Key to the Maintenance Problem", Butterworth Journal of Information and Software Technology, January/February 1991
- ⁶Rombach, H. D. and V. R. Basili, "Quantitative Assessment of Maintenance: An Industrial Case Study", Proceedings From the Conference on Software Maintenance, September 1987
- ⁶Rombach, H. D. and L. Mark, "Software Process and Product Specifications: A Basis for Generating Customized SE Information Bases", Proceedings of the 22nd Annual Hawaii International Conference on System Sciences, January 1989
- ⁷Rombach, H. D. and B. T. Ulery, Establishing a Measurement Based Maintenance Improvement Program: Lessons Learned in the SEL, University of Maryland, Technical Report TR-2252, May 1989
- ¹⁰Rombach, H. D., B. T. Ulery, and J. D. Valett, "Toward Full Life Cycle Control: Adding Maintenance Measurement to the SEL", Journal of Systems and Software, May 1992
- ¹⁴Seaman, C. B., V. R. Basili, "Communication and Organization in Software Development: An Empirical Study", University of Maryland, Computer Science Technical Report, CS-TR-3619, UMIACS-TR-96-23, April 1996
- ¹⁵Seaman, C. B, V. R. Basili, "An Empirical Study of Communication in Code Inspection", Proceedings of 19th International Conference on Software Engineering (ICSE-19), May 1997, pp. 96-106
- ¹⁶Seaman, C. B and V. R. Basili, "An Empirical Study of Communication in Code Inspections", University of Maryland,
- ¹⁶Seaman, C. B, V. R. Basili, "Communication and Organization in Software Development: An Empirical Study", IBM Systems Journal, Vol. 36, No. 4, 1997
- ¹⁶Seaman, C. B, V. R. Basili, The Study of Software Maintenance Organizations and Processes
- ⁶Seidewitz, E., "Object-Oriented Programming in Smalltalk and Ada", Proceedings of the 1987 Conference on Object-Oriented Programming Systems, Languages, and Applications, October 1987
- ⁵Seidewitz, E., "General Object-Oriented Software Development: Background and Experience", Proceedings of the 21st Hawaii International Conference on System Sciences, January 1988
- ⁶Seidewitz, E., "General Object-Oriented Software Development with Ada: A Life Cycle Approach", Proceedings of the CASE Technology Conference, April 1988
- ⁹Seidewitz, E., "Object-Oriented Programming Through Type Extension in Ada 9X", Ada Letters, March/April 1991
- ¹⁰Seidewitz, E., "Object-Oriented Programming With Mixins in Ada", Ada Letters, March/April 1992
- ¹²Seidewitz, E., "Genericity versus Inheritance Reconsidered: Self-Reference Using Generics", Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, October 1994
- ⁴Seidewitz, E. and M. Stark, "Towards a General Object-Oriented Software Development Methodology", Proceedings of the First International Symposium on Ada for the NASA Space Station, June 1986
- ⁹Seidewitz, E. and M. Stark, "An Object-Oriented Approach to Parameterized Software in Ada", Proceedings of the Eighth Washington Ada Symposium, June 1991

- ⁸Stark, M., "On Designing Parametrized Systems Using Ada", Proceedings of the Seventh Washington Ada Symposium, June 1990
- ¹¹Stark, M., "Impacts of Object-Oriented Technologies: Seven Years of SEL Studies", Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, September 1993
- ⁷Stark, M. E. and E. W. Booth, "Using Ada to Maximize Verbatim Software Reuse", Proceedings of TRI-Ada 1989, October 1989
- ⁵Stark, M. and E. Seidewitz, "Towards a General Object-Oriented Ada Lifecycle", Proceedings of the Joint Ada Conference, March 1987
- ¹⁵Stark, M., "Using Applet Magic (tm) to Implement an Orbit Propagator: New Life for Ada Objects", Proceedings of the 14th Annual Washington Ada Symposium (WAdaS97), June 1997
- ¹³Stark, M. and E. Seidewitz, "Generalized Support Software: Domain Analysis and Implementation", Addendum to the Proceedings OOPSLA '94, Ninth Annual Conference, Portland, Oregon, U.S.A., October 1994, pp. 8 - 13
- ¹⁰Straub, P. A. and M. V. Zelkowitz, "On the Nature of Bias and Defects in the Software Specification Process", Proceedings of the Sixteenth International Computer Software and Applications Conference (COMPSAC 92), September 1992
- ⁸Straub, P. A. and M. V. Zelkowitz, "PUC: A Functional Specification Language for Ada", Proceedings of the Tenth International Conference of the Chilean Computer Science Society, July 1990
- ⁷Sunazuka, T. and V. R. Basili, Integrating Automated Support for a Software Management Cycle Into the TAME System, University of Maryland, Technical Report TR-2289, July 1989
- ¹³Thomas, W. M., A. Delis, and V. R. Basili, An Analysis of Errors in a Reuse-Oriented Development Environment, University of Maryland, Computer Science Technical Report, CS-TR-3424, UMIACS-TR-95-24
- ¹⁶Tesoriero, R., M. Zelkowitz, A Model of Noisy Software Engineering Data (Status Report), Proceedings of the Twentieth International Conference on Software Engineering, April 19 - 25, 1998
- ¹⁶Tesoriero, R., M. Zelkowitz, WEBME: A Web-based tool For Data Analysis and Presentation, IEEE Internet Computing
- ¹⁶Thomas, W. M., A. Delis, and V. R. Basili An Analysis of Errors in a Reuse-Oriented Development Environment, Journal of Systems Software, 38:211 - 224, 1997
- ¹⁰Tian, J., A. Porter and M. V. Zelkowitz, "An Improved Classification Tree Analysis of High Cost Modules Based Upon an Axiomatic Definition of Complexity", Proceedings of the Third IEEE International Symposium on Software Reliability Engineering (ISSRE 92), October 1992
- Turner, C. and G. Caron, A Comparison of RADC and NASA/SEL Software Development Data, Data and Analysis Center for Software, Special Publication, May 1981
- ¹⁰Valett, J. D., "Automated Support for Experience-Based Software Management", Proceedings of the Second Irvine Software Symposium (ISS _92), March 1992
- ⁵Valett, J. D. and F. E. McGarry, "A Summary of Software Measurement Experiences in the Software Engineering Laboratory", Proceedings of the 21st Annual Hawaii International Conference on System Sciences, January 1988
- ¹⁴Waligora, S., J. Bailey, and Mike Stark, "The Impact of Ada and Object-Oriented Design in NASA Goddard's Flight Dynamics Division", July 1996
- ³Weiss, D. M. and V. R. Basili, "Evaluating Software Development by Analysis of Changes: Some Data From the Software Engineering Laboratory", IEEE Transactions on Software Engineering, February 1985
- ⁵Wu, L., V. R. Basili, and K. Reed, "A Structure Coverage Tool for Ada Software Systems", Proceedings of the Joint Ada Conference, March 1987

- ¹Zelkowitz, M. V., "Resource Estimation for Medium-Scale Software Projects", Proceedings of the Twelfth Conference on the Interface of Statistics and Computer Science. New York: IEEE Computer Society Press, 1979
- ²Zelkowitz, M. V., "Data Collection and Evaluation for Experimental Computer Science Research", Empirical Foundations for Computer and Information Science (Proceedings), November 1982
- ⁶Zelkowitz, M. V., "The Effectiveness of Software Prototyping: A Case Study", Proceedings of the 26th Annual Technical Symposium of the Washington, D.C., Chapter of the ACM, June 1987
- ⁶Zelkowitz, M. V., "Resource Utilization During Software Development", Journal of Systems and Software, 1988
- ⁸Zelkowitz, M. V., "Evolution Towards Specifications Environment: Experiences With Syntax Editors", Information and Software Technology, April 1990
- ¹⁴Zelkowitz, M. V., "Software Engineering Technology Infusion Within NASA", IEEE Transactions On Engineering Management, vol. 43, no. 3, August 1996

NOTES:

- ¹This article also appears in SEL-82-004, *Collected Software Engineering Papers: Volume I*, July 1982.
- ²This article also appears in SEL-83-003, *Collected Software Engineering Papers: Volume II*, November 1983.
- ³This article also appears in SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985.
- ⁴This article also appears in SEL-86-004, *Collected Software Engineering Papers: Volume IV*, November 1986.
- ⁵This article also appears in SEL-87-009, *Collected Software Engineering Papers: Volume V*, November 1987.
- ⁶This article also appears in SEL-88-002, *Collected Software Engineering Papers: Volume VI*, November 1988.
- ⁷This article also appears in SEL-89-006, *Collected Software Engineering Papers: Volume VII*, November 1989.
- ⁸This article also appears in SEL-90-005, *Collected Software Engineering Papers: Volume VIII*, November 1990.
- ⁹This article also appears in SEL-91-005, *Collected Software Engineering Papers: Volume IX*, November 1991.
- ¹⁰This article also appears in SEL-92-003, *Collected Software Engineering Papers: Volume X*, November 1992.
- ¹¹This article also appears in SEL-93-001, *Collected Software Engineering Papers: Volume XI*, November 1993.
- ¹²This article also appears in SEL-94-004, *Collected Software Engineering Papers: Volume XII*, November 1994.
- ¹³This article also appears in SEL-95-003, *Collected Software Engineering Papers: Volume XIII*, November 1995.
- ¹⁴This article also appears in SEL-96-001, *Collected Software Engineering Papers: Volume XIV*, October 1996.
- ¹⁵This article also appears in SEL-97-002, *Collected Software Engineering Papers: Volume XV*, October 1997.
- ¹⁶This article also appears in SEL-99-002, *Collected Software Engineering Papers: Volume XVI*, November 1999.